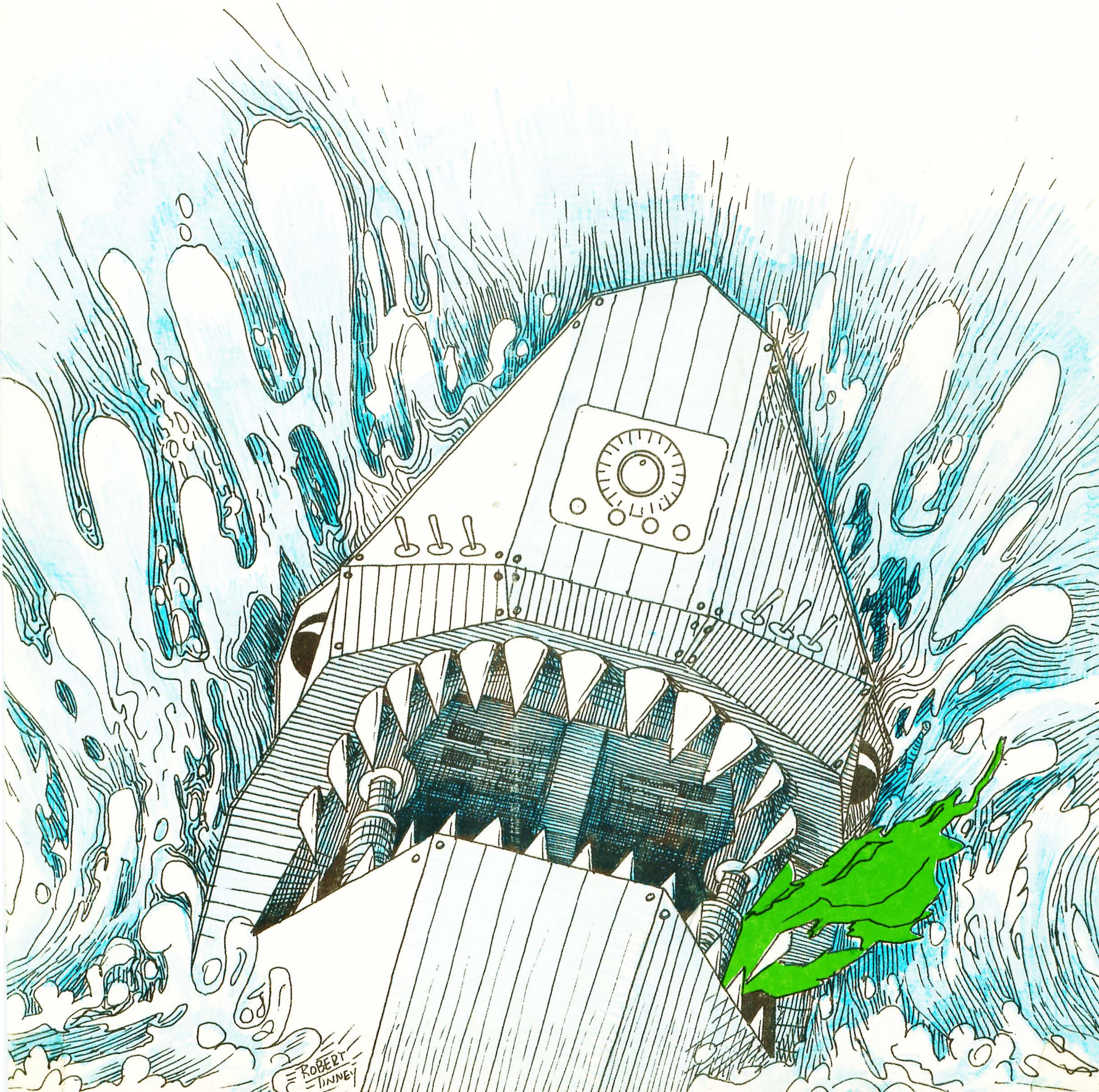
APRIL 1976 $1.50 (12 bits)

BYTE

the small systems journal



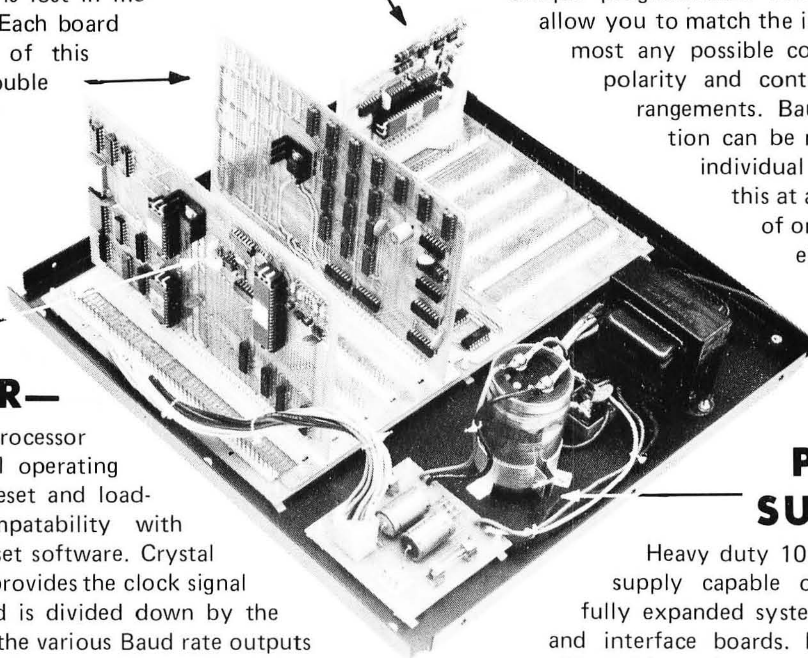
WHY SETTLE FOR LESS— THAN A COMPLETE 6800 SYSTEM

MEMORY—

All static memory with selected 2102 IC's allows processor to run at its maximum speed at all times. No refresh system is needed and no time is lost in memory refresh cycles. Each board holds 4,096 words of this proven reliable and trouble free memory. Cost—only $125.00 for each full 4K memory.

INTERFACE—

Serial control interface connects to any RS-232, or 20 Ma. TTY control terminal. Connectors provided for expansion of up to eight interfaces. Unique programmable interface circuits allow you to match the interface to almost any possible combination of polarity and control signal arrangements. Baud rate selection can be made on each individual interface. All this at a sensible cost of only $35.00 for either serial, or parallel type



PROCESSOR—

"Motorola" M6800 processor with Mikbug® ROM operating system. Automatic reset and loading, plus full compatability with Motorola evaluation set software. Crystal controlled oscillator provides the clock signal for the processor and is divided down by the MC14411 to provide the various Baud rate outputs for the interface circuits. Full buffering on all data and address busses insures "glitch" free operation with full expansion of memory and interfaces.

POWER SUPPLY—

Heavy duty 10.0 Amp power supply capable of powering a fully expanded system of memory and interface boards. Note 25 Amp rectifier bridge and 91,000 mfd computer grade filter capacitor.

DOCUMENTATION—

Probably the most extensive and complete set of data available for any microprocessor system is supplied with our 6800 computer. This includes the Motorola programming manual, our own very complete assembly instructions, plus a notebook full of information that we have compiled on the system hardware and programming. This includes diagnostic programs, sample programs and even a Tic Tac Toe listing.

Mikbug® is a registered trademark of Motorola Inc.

SwTPC 6800
Computer System
with serial interface and 2,048 words of memory. $395.00



☐ Enclosed is $395 for my SwTPC Computer Kit ☐ Send Data

☐ or BAC ______________ # ______________

☐ or MC ______________ Ex Date ______________

NAME ______________

ADDRESS ______________

CITY ______________ STATE ______________ ZIP ______________

Southwest Technical Products Corp., Box 32040, San Antonio, Texas 78284

CCD I^2L EPROM

PLA DIP DFT

ED

LSI

DTL

ECL

CAN YOU DEFINE THE ABOVE MICROCOMPUTER TERMS?

For an authoritative up-to-the minute source of microcomputer terminology and definitions . . . send for this

NEW MICROCOMPUTER DICTIONARY!

TITLE: *Microcomputer Dictionary & Guide*
AUTHOR: *Charles J. Sippl and David Kidd*

This new microcomputer dictionary fills the urgent need for all computer people, engineers, scientists, industrialists, communications people — as professionals, amateurs, teachers, or students — to become quickly acquainted with the terminology and nomenclature of a new revolution in computer control capabilities in areas that pervade most of man's daily activities.

Over 8000 definitions and explanations of terms and concepts (704 pages) relating to microprocessors, microcomputers and microcontrollers. There are also separate appendices on: programmable calculators; math and statistics definitions, flowchart symbols and techniques; binary number systems and switching theory; symbol charts and tables; summaries of BASIC FORTRAN and APL. In addition there is a comprehensive electronics/computer abbreviations and acronyms section. Price: **$17.95**

(Also available from technical book stores and selected electronic distributors.)

MATRIX PUBLISHERS, INC.
Dept. BM, 207 Kenyon Road, Champaign, IL 61820

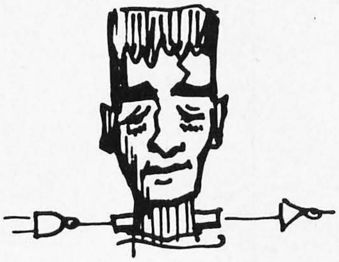
Please send me the new MICROCOMPUTER DICTIONARY under your 15 day no risk trial guarantee. If payment accompanies order we pay all shipping and handling charges. (Ill. customers add 5% sales tax)

Name ______________________

Address ______________________

City ______________ State ______________ Zip

In This BYTE



Curiosity, experimentation and imagination are great aids to the user of a computer system. Charles A Crayne describes two instruction set variations he found while **Programming the Implementation** of an 8008 processor as a SCELBI 8H minicomputer.

Can a computer predict your state of mind? A better question would be: Is there a theory which can be computationally verified to predict your state of mind? **Biorhythm for Computers** is an article by Joy and Richard Fox on the use of a BASIC calculation to provide predictions based upon the pseudo science of a simple biorhythm hypothesis.

A key element of computing is the expression of ideas in the form of programs and algorithms. It takes the **Magic of Languages** to make such expressions in a form the computer can understand. Turn to Theodor Nelson's article for an introduction to some basic concepts of computer languages.

Ho hum, another memory article? Not quite. When Don Lancaster adds a twist of ingenuity, you find out **How to Build a Memory With One Layer Printed Circuits**, saving the trouble of using many wire jumpers or figuring out how to make double layer boards in the kitchen PC laboratory.

Aargh! (or, How to Automate PROM Burning Without EML) was Peter Helmers' reaction to a suggestion that relays be used to control a circuit. Study his figures for technical information and learn about the exciting new field of EML[sic] logic in the accompanying text.

It could not take long to find another contender in the marketplace for compact computer systems. In this issue, J Bradley Flippin discusses **The SR-52: Another "World's Smallest Computer System."**

Input is often done with a switch contact when simple on off states of electromechanical systems are considered. But suppose you want to program stage lighting or drive a keyboard machine with solenoids. Then you'll need information on **Controlling External Devices with Hobbyist Computers**. Robert J Bosen presents some ideas on the subject.

Computers require a large dose of that arcane art, interfacing. Jay A Cotton shows one example of that art in his discussion of how to **Interface an ASCII Keyboard to a 60 mA TTY Loop**.

Introspection is a prime technique for analyzing the human consciousness. Many parallels can be drawn between the design of complicated computer networks and knowledge of human mental functions. While not purporting to be a complete or final model of human mental functions, Joe Murray's article on **Frankenstein Emulation** provides some good inputs on the ultimate hobby: modeling human behavior.

A Note on the Current BYTE

Last month, the March issue of BYTE had a theme of magnetic recordings for digital data: four articles in that issue concerned various aspects of mass storage on the magnetic medium. This month, the theme is somewhat nebulous and is summed up by "April Fool!". The cover and several items in this issue are on a theme of far out applications and fun. I'll leave it to readers to figure out what items other than the cover fall into this category; however, all articles in this issue are instructive and informative, even those which are consistent with the theme. **CH**

Minicomputers and microcomputers are really quite similar. The former are simply faster and more expensive than the current versions of the latter. Thus when you **Design an On Line Debugger** for a minicomputer, as Robert Wier and James Brown have done, the same general interactive design can be used as the control panel interface for any microcomputer as well. Add an on line debugger to your computer and you'll make it a much easier device to use.

What's better than an 8 bit processor in a 40 pin package? Why, a 16 bit processor in a 64 pin package, of course. In his **Microprocessor Update: Texas Instruments TMS9900**, Robert Baker provides readers with an overview of this exciting new computer which is sure to find its way into personal computing systems over the next year or so.

Ingenuity is an old American tradition. Roger W Thompson makes his contribution to that tradition in his description of how to **Save Money Using Mini Wire Wrap**, a socketless penny pinching way of wiring integrated circuit projects.

On the cover, at the suggestion of Tully Peschke, Robert Tinney created a fantasy on a theme of BYTE. First aid has already been applied and it is expected that the banner will be fixed in time for the May issue.

In the Queue

Foreground

Background

Nucleus

BYTE #8
APRIL 1976

staff

PUBLISHERS
Virginia Peschke
Manfred Peschke
EDITOR
Carl T Helmers, Jr
GENERAL MANAGER
Manfred Peschke
PRODUCTION MANAGER
Judith Havey
PRODUCTION ASSISTANT
Elizabeth Alpaugh
CIRCULATION
Deborah R Luhrs
PUBLISHERS ASSISTANTS
Cheryl Hurd
Carol Nyland
Deena Zealy
ADVERTISING
Elizabeth Alpaugh
Virginia Peschke
TYPOGRAPHY
Custom Marketing Resources, Inc
Goodway Graphics
Mary Lavoie
Taimi Woodward
PHOTOGRAPHY
Ed Crabtree
Custom Marketing Resources, Inc
ART
Bill Morello
PRINTING
Custom Marketing Resources, Inc
The George Banta Company
ASSOCIATES
Dan Fylstra
Don Lancaster
Harold A Mauch
Chris Ryland

BYTE magazine is published monthly by BYTE Publications, Inc , 70 Main St, Peterborough, New Hampshire 03458. Subscription rates are $12 for one year worldwide. Two years, $22. Three years, $30. Second class postage application pending at Peterborough New Hampshire 03458 and at additional mailing offices. Phone 603-924-7217.

Address editorial correspondence to Editor, BYTE, 70 Main St, Peterborough NH 03458.

Customization -- The Expression of Individuality

Editorial by Carl Helmers

Each person who gets involved in this activity of acquiring and using a personal computer system will sooner or later appreciate the results of customization. Each computer system comes out differently according to the tastes and creativity of its owner (to say nothing of his or her pocketbook). The number "n" of choices available in creating a personal system is large, and is getting larger every day.

Individuality of Systems

Each personal computing system is an individual creation, despite the mass production genesis of all the parts and pieces. To understand this at a global level, simply consider the options available in the choice of standard microprocessor designs which make up the starting point for any system.

At the present time, there are kits on the market for half a dozen or so different microcomputer designs. The first expression of individuality you have to make is in your choice of the microprocessor design for your system. Choosing one such processor puts you into a group of people employing the same instruction set, a group which can share software experiences directly. These users are distinguished from the logical grouping of users associated with the other microprocessor architectures. (Note that the "user groups" for personal computing are logically a result of CPU choice since all users of the same chip design share common problems independent of the means of wiring up and packaging the computer.)

Having limited yourself to one particular microprocessor, the next level of exercising individuality in creating a system is your choice of a means to package and implement a processor using that CPU. Here again, the options are several. Some individuals "home brew" the packaging and system design with wire wrap or other interconnection techniques. Others elect to purchase a pre-engineered kit of parts from one of BYTE's advertisers. Whatever the choice of a basic system assembly with the processor, there are numerous system design options available to you. How much memory? What type of interactive display terminal? Will the system use audio tape or will it have a floppy disk as mass store? Does the supplier have a BASIC package? Does the supplier make an assembler, monitor and text editor available? Assuming a particular set of answers to these questions, you will end up with a fairly unique system, but one possibly identical at the start to the systems of several other individuals who use the same kit maker. What is the final step in customization?

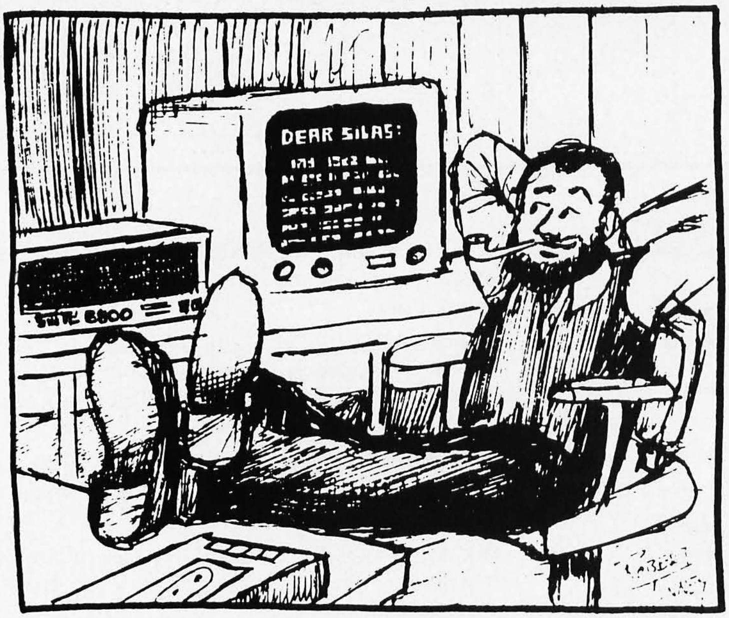
Modularity and Software

The customization of a system is ultimately achieved through the variations in the optional hardware modules attached to it, and through the personal library of software built up by the system's owner and user.

The customization of hardware starts in the choice of options available either from the manufacturer of the kit or from independent suppliers. There is presently only one family of compatible peripherals, those employing the MITS Altair backplane interconnection conventions. (For the present time, in order to have the widest choice of prepackaged component kits an Altair compatible bus is virtually a necessity.) If you purchase an Altair incompatible system, it will prove helpful to home brew an Altair compatible bus extension for interfacing such peripherals; if you engineer a home brew system, such an extension is also useful. (That is one project in my queue for my own home brew 6800 system.)

Once you have a basic system up and running, the customization of hardware continues in the choice of specific peripherals for applications, and in the choice of add on memory and hardware to augment the system as you continue to use it. Custom peripherals are required wherever a process control or sensing application is involved. If the computer is to control a model railroad, for instance, you'll have to custom design

continued on page 94



Letters

STAR TREK AND SPACE WAR FORUM

A number of readers have expressed interest in the use of personal computing equipment for STAR TREK and other space war games. Here is a collection of letters on the subjects. Continuation of the discussion and accounts of personal activities in this area are encouraged.

Members of the staff have been known to arrive at work two hours early in order to get their morning "fix."

OREGON STATE FORTRAN STAR TREK

In reply to the letters by Richard Wexler and Stewart Shelton regarding Star Trek and Spacewar, a version of Star Trek written in FORTRAN IV is currently being run on the CDC Cyber, under KRONOS 2.1, at the Oregon State University Computer Center, Corvallis OR 97331. The Spacewar to which Mr Shelton refers is, I believe, that which was publicized during the "National Spacewar Competition." My source of info on this was the *DEC EduGram*. I don't have access to this any more, but I believe it was in a late 1971 or a 1972 issue that the game was first described. DEC (Maynard MA) may be able to tell you more about this particular version. I am familiar with at least three other Spacewar games and would like to hear from other Spacewar freaks about different versions they might know of.

I've thoroughly enjoyed every issue of BYTE and can hardly wait for each new one. The only thing that bothers me is the lack of articles on programming theory. Will there be any sort of series covering such things as searching and sorting, parsing, tree structures, etc.?

Chip Weems
728 Mildred Ln SE
Salem OR 97302

LARGE SCALE STAR TREK

. . . There has been considerable discussion of Star Trek (Spacewar) in the January issue. I'd like to pass on some information regarding the version of Star Trek we have currently running on a Data General Eclipse computer at work.

The genesis of the FORTRAN IV Star Trek program we have is unknown. We got it from a friend who got it from a friend who . . . . As far as I could tell, the code was Xerox FORTRAN although I physically got it on mag tape from a Varian 620i. The source code of the version we received was 1800 statements. Some effort was required for program transition to DG FORTRAN. We also spent considerable time debugging and cleaning up the code.

Our full program requires 25 K (16 bit words) of core for execution. We have 48 K of memory and operate background and foreground. Since the operating system (RDOS) takes about 16 K, any applications program over 16 K either steals the machine or severely limits the ability of the machine to support two users. (Particularly at lunch time when we make the machine available to computer games players.) Therefore, we broke the program into overlays to obtain a load module of 16 K. This works fine, but the disk sure rattles when torpedoes start flying. However, two players may play individual games at the same time. A further refinement was to compile the DG's FORTRAN 5 which significantly speeds execution and results in some saving of core (14 K).

This version of Star Trek is quite complex, although versions for large machines are even more complex. The game randomly determines the number of Romulans and Klingons in the Unholy Alliance and generates a reasonable number of Stardates in which to complete the mission. Automatic scoring of player proficiency is built in, along with appropriate insulting messages for those unfortunates who hit a star or meet their demise through other more sophisticated means. This game is not for the occasional player as a certain amount of skill is needed for full enjoyment. The program operates in simulated real time so the player with slow reactions is penalized.

Star Trek is an extremely addictive computer game. Members of the staff have been known to arrive at work two hours early in order to get their morning "fix." While most people quickly learn how to beat the general simple computer game and become bored with that game, very few can beat Star Trek with any regularity. No players have been

known to become bored over an extended period of time.

Since such an extensive game as we have available is not useful to the average hobbyist on his micro, it seems reasonable that many will turn to one of the BASIC versions of Space War such as you have referenced as being available in the DEC book of 101 BASIC Games. Your January book review failed to mention that the reproduction quality of programs in that book varies for each of the programs. Unfortunately, the reproduction of Space War in that book is very hard, if not impossible to read, because the master was obtained from a Teletype that was not properly aligned. The bottoms of the characters are cut off in the copy of the book that I bought. However, most of the programs in that book are reproduced quite well and I would add my general endorsement to your book review.

It's like being pioneers—spirits are high and there is a wilderness to contend with.

I enjoy BYTE and would add my vote to others you have received regarding the happy combination of ham radio and computers as the source of future articles.

Dave Vandaveer
Santa Ana CA

SOME SPACE WAR BACKGROUND

Regarding the enquiry concerning Space War (page 86, January BYTE) I can agree that it is a stimulating, dynamic (and therefore very time-consuming) game. Its origins are shrouded in mystery, but it is part of a standard demonstration packet released by DEC for their machines with graphics capability, such as the PDP-12 or the GT40 system. The essence of the game is the control of two rockets (displayed on a CRT)which have missile-firing systems; the object is to blast your dreaded enemy from the spaceways. Most versions use four bit-controlled parameters for each ship: rotate up, rotate down, fire thruster and fire missiles. More elaborate versions include meteorite obstacles, a central sun which modifies trajectories due to its gravitational pull and "space warp," the ability to disappear when threatened with incipient destruction (but not know when or where you will reappear).

The most primitive version of the game will run in a PDP-8 with 4 K. There must be some provision for graphics display, either a point-plotter or DA converters and a CRT. Control can be via console switches, and timing is aided by a real time clock, although this latter requirement is not necessary. The main problem with running the game on a micro is word size. Velocity and position of the rockets is calculated iteratively, and small roundoff errors yield large *net changes* as the game goes on. For this reason, all parameters are calculated double precision (24 bit) on the PDP-8, which is triple precision on most micros. The increased overhead as a result of this requirement may spoil the timing and slow the game down, although the PDP-8 executes a complete update loop in somewhat less than 10 ms, and this can be slowed to about 30 ms before the onset of noticable flicker. The display may pose a problem as well, since 10 bit resolution on a CRT is acceptable, but 8 bit may be pushing things a bit.

A B Bonds, PhD
Berkeley CA

NAVIGATING AN OCEAN OF INFORMATION

First, let me say that I'm enjoying BYTE very much and look forward to each new issue.

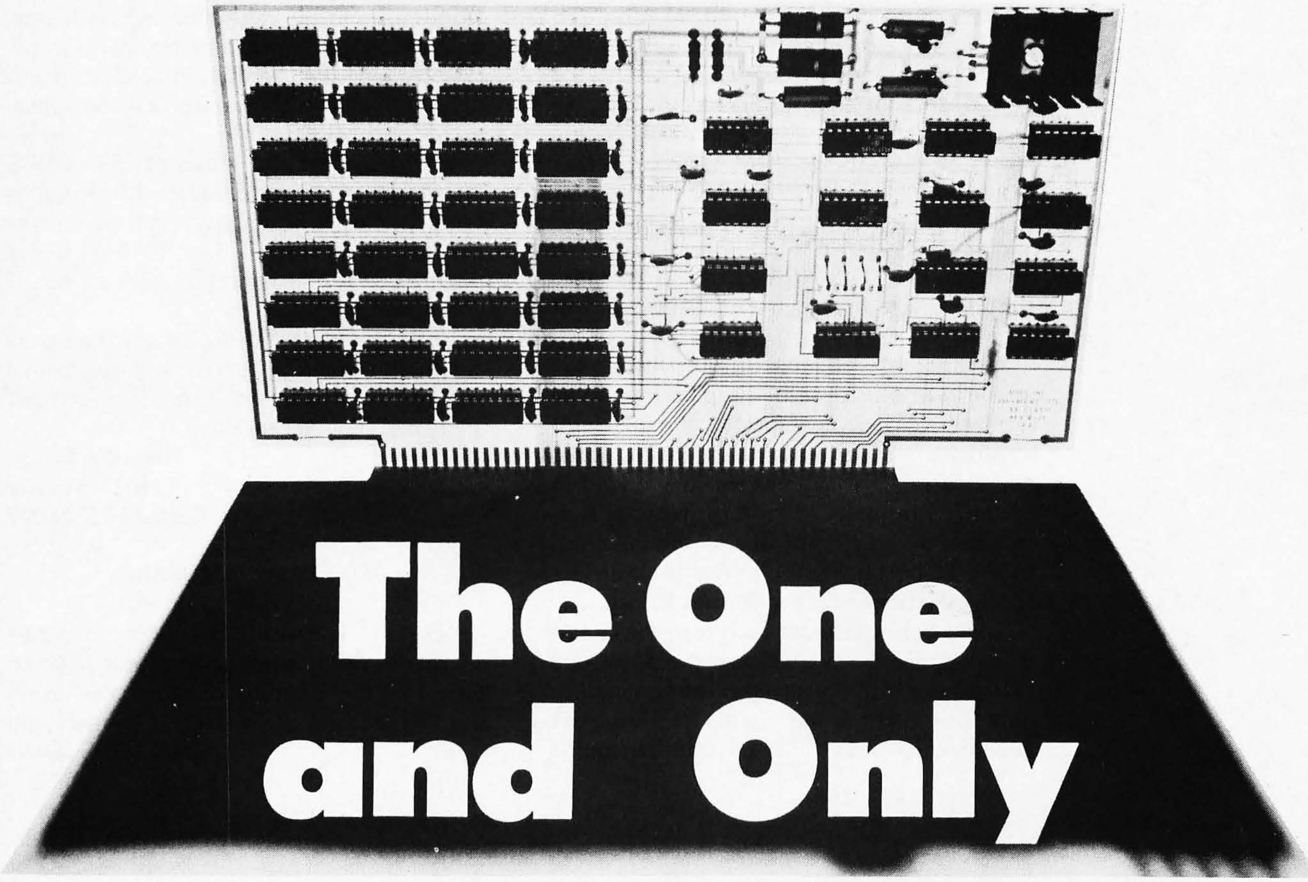
The microcomputer system I've built uses an 8008 but will eventually expand to symbiosis with an 8080 CPU. I want to express my happiness that you are not abandoning those of us hobbyists who have 8008 based machines. I know all too well the limitations of the 8008, but I think it is still a superb way to start out. The 8008 instruction set almost immediately inspires affection with its non-threatening three letter mnemonics and their nice correlation to octal number representation.

I think most of us know that we're into a unique thing with this burgeoning new baby of a hobby. It's like being pioneers – spirits are high and there is a wilderness to contend with. Big time centralized computers were not a panacea for the world's ills. The new wilderness we face is information glut and it threatens to swamp us more essentially at the individual level than at the institutional level. Microcomputers are a new tool that promise deliverance. They can be a seemingly playful tool, but they possess the power to navigate that ocean of information which modern man must sail upon. Well, I just wanted to philosophize a little and say that I think this new hobby (and BYTE) will be an aid to us all.

Adam Trent
Ascension Island
Patrick AFB FL

AN IMPROVED CASSETTE INTERFACE CIRCUIT

As might be expected, we have made some changes and improvements to our cassette interface circuit which will be of



Altair® 4K Static.

Altair 4K Static from MITS is unquestionably the finest 4K static memory available anywhere. It is also the fastest.

Altair 4K Static uses Intel 2102 A-4 memory chips which have a worst case access of 450 nanoseconds at 70 °C. At normal system temperatures the access times are typically less than 300 nanoseconds.

Altair 4K Static is fully isolated from the system bus by Schmitt ™ Triggers. Thus, the excessive capacitive loading caused by other 4K static memories is eliminated. Use of these triggers on all Altair 4K static inputs greatly reduces noise. Internal data collection nodes also use Schmitt Triggers, which prevents internal data bus noise from being transmitted to the system data bus.

Altair 4K Static is the only 4K static supported by MITS. Owners of Altair 4K Static are eligible to qualify for discounts on AltairBASIC and other MITS products.

Altair 4K Static is the only 4K static that comes with all the required Altair hardware including edge connectors and card guides.

Altair 4K Static is the answer for Altair owners who need static memory for special applications such as the TV Dazzler from Cromemco.

PRICES:

Altair 4K Static Kit **$159**
Altair 4K Static Kit with 2K Memory **$134**
Chip set to convert 2K to 4K . . . . **$ 45**

SPECIAL— Altair Documentation Notebook. Contains catalog, price sheet, Computer Notes newspaper, Software Information Package, technical data on Altair hardware, list of authorized Altair dealers, list of computer clubs, survey of home computing market, and much more. All in top quality three ring binder. Only $5 plus $1 for postage and handling. Offer expires April 30, 1976.

MAIL THIS COUPON TODAY

Enclosed is check for $ ____________
BankAmericard # ____________ or Master Charge # ____________
☐ Altair 4K Static ☐ Altair 2K Static
Include $3 for postage and handling
☐ Please send me the Altair Documentation Notebook. Enclosed is $5 plus $1 for postage and handling.
☐ Please send free catalog and price sheet.

NAME ____________
ADDRESS ____________
CITY ____________ STATE & ZIP ____________

MITS/2450 Alamo SE/Albuquerque, NM 87106/505-262-1951

Prices, delivery and specifications subject to change.

mits

2450 Alamo SE/Albuquerque, NM 87106/505-262-1951

interest to your readers. These improvements were still being worked out when I submitted the article ("Digital Data on Cassette Recorders," page 40, March 1976 BYTE), hence they were not included.

We were quite disappointed in the performance of the 4047 as a retriggerable one shot, therefore we went back to the 9601 or 74122 we had used in earlier designs. We have also eliminated the need for ± 12 volt power supplies by changing the design of the signal conditioner circuit and by using the AY5-1014 UART. The entire circuit now operates from a single 5 volt supply.

We discovered the circuit described in the article was sensitive to the polarity of the received signal from tape. To eliminate this sensitivity we went back to a double edged pulse former. This required the one shot period to be reduced to 278 microseconds and the elimination of one of the flip flops in the Phase Locked Loop feedback divider string. Adjustment is the same as before.

Some hobbyists who have seen the circuit have wondered why we used the 339 Quad Comparator for the signal conditioner since only one section was used. The Pronetics circuit card uses one of the remaining sections as a level sensor to inform the user of proper playback level. Another section is used to indicate when data is being received from tape and the remaining section is used to drive a relay which remotely controls the cassette tape unit. If these functions are not necessary a 311 comparator can be substituted for the 339.

We designed out most of the CMOS integrated circuits because of handling problems in low humidity (static discharge destruction of gates). The 4046 PLL was retained to permit operation with a single 5 volt power supply.

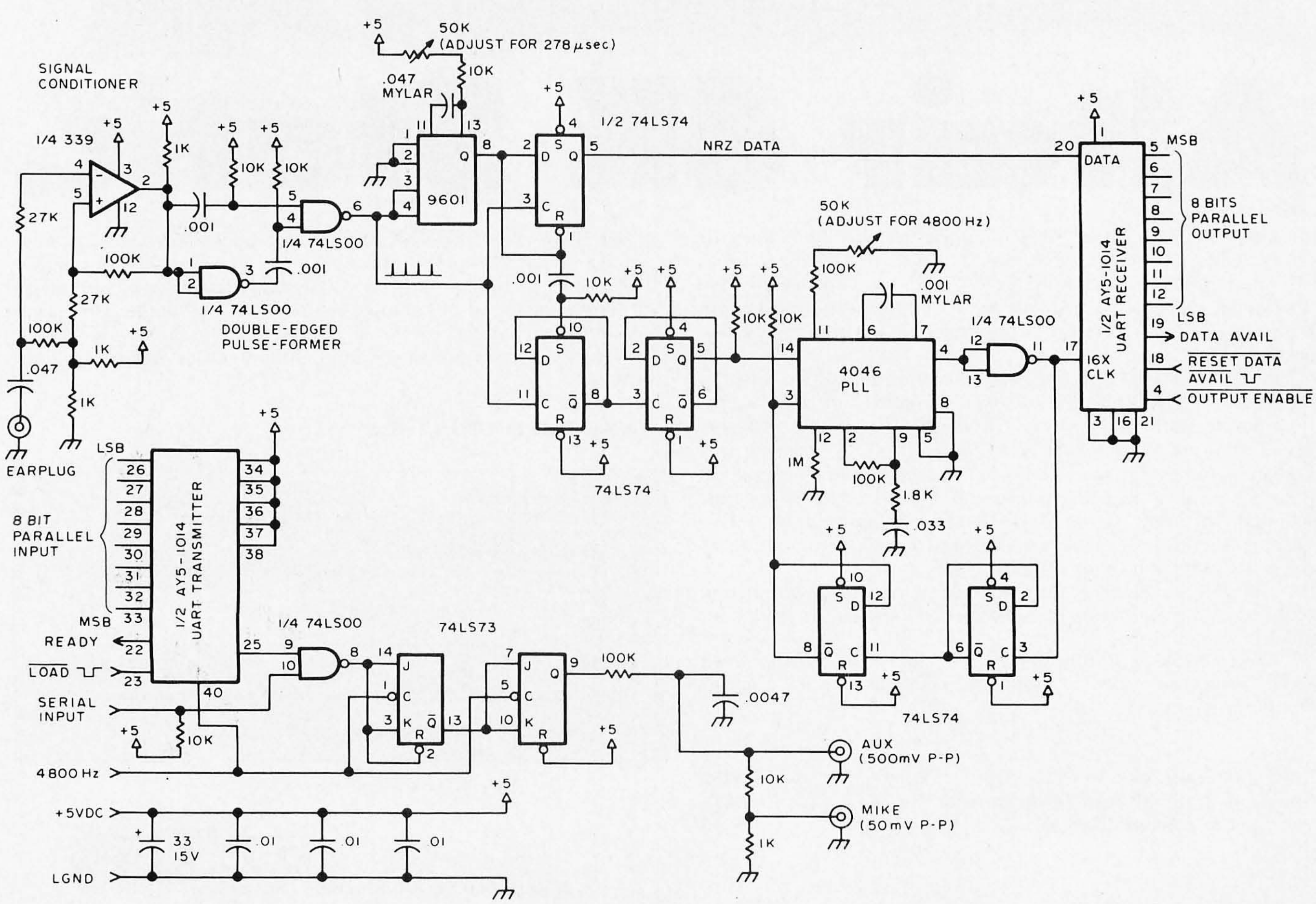
Enclosed is a drawing of the improved circuit. If your readers have any questions I will be happy to answer if they include a self addressed stamped envelope.

Harold A Mauch
4021 Windsor
Garland TX 75042

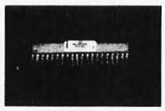
RTTY AND MORSE

So far, I have found your magazine excellent in all respects. You appear to be on top of all the latest machines and devices on the market, and the articles are timely and

continued on page 71



You can buy this microcomputer for $39.95, but...



MC 6800

We would be a bit surprised if you could do anything meaningful without additional hardware and software. **Wave Mate's Jupiter II**™ isn't the kind of microcomputer kit you only stare at... when you've completed your Jupiter II just plug in your terminal and you're ready to go. That's because it goes beyond the sum of its high quality parts. It's the ultimate micro kit experience. **In performance, in documentation, in reliability.** First, consider its superb features. It has small **pluggable wire wrapped cards** easily tailored to suit your modifications. **Every IC is socketed** and **100% burn-in tested.**

In fact every part including the powerful MC 6800 CPU and the 8K dynamic RAM is guaranteed for 120 days. It has the best software around, **System Monitor** and **Debug programs** (ROM). Includes powerful **text editor** and **Motorola compatible assembler.** And BASIC at no extra cost. Because we've been making microcomputer systems for over 4 years, we can offer you the broadest line of interfaces including TV terminal and dual audio cassette. Impressive. And yet the grandest feature is the experience of completing a kit that works. Guaranteed.

Wave Mate

ACT NOW AND SAVE. (Good until April 1, 1976)
SPECIAL price $1445.00 (assembled $1950.00)

Jupiter II Microcomputer Kit Includes:

- Modular plug-in power supply
- 9 module PC backplane
- CPU module
- System monitor module
- 8K dynamic RAM module
- Serial RS 232 communication interface module
- Front panel module
- Front panel
- Wire; cut, stripped, color coded
- Rack mount module cage
- Wire wrap tool
- Wire unwrap tool
- Cables, connectors, all other necessary hardware
- Software (editor, debug, assembler, BASIC)
- Assembly manuals
- Operators manuals
- Theory of operation manuals
- Annual membership in users group



☐ Send Jupiter II micro kit.
☐ Kit ☐ Assembled
☐ Send details on kit-a-month
☐ I'm not convinced; send me 101 reasons why.
☐ Please send free Jupiter II System catalogue.

☐ Check enclosed for $ ______
Include $10.00 for postage and handling. California residents add 6% tax Delivery 60 days ARO

Name ______
Address ______
City/State/Zip ______

WAVE MATE
1015 West 190th Street Gardena, California 90248
Telephone (213) 329-8941

Wave Mate

WARRANTY
120 days on all parts, materials and workmanship.

April Fool came a little early with Space Ace. We aced out on Bob Baker's puzzle in March BYTE, page 77, by omitting the word list. Here it is again, with the word list. Answers will appear in May BYTE.

N	F	S		B	R			T		N		S			P
	R		C		R	D		M		N	S			N	R
T			T	P		T	R		T		R		P		
W	R		T		X	T		R	N		L	R			G
	D		C		D		C	N					N	S	R
	L	B			R		V	S	G		B	W	C		
C				D	N		P		N	T	R		H	C	M
N	T	N	M	R	T		C					N			H
	C	T		P	C			C	M	G	R	D	N	R	T
L			T		L	S	R				Y	N	T	P	
	M	G			S		N	L	L	T	N		R	P	R
V	P		D		D	P	C	L			L	F	Y		
	L	R	R	L					S	L			R	R	G
		P		T	Y	P			T		N		B	R	
Q	X		S	C		L		R	L		T		R		L
	F		N	C	T			N	G		S	S		Y	L

Space Ace

Robert Baker
34 White Pine Dr
Littleton MA 01460

By inserting the missing vowels (a, e, i, o, and u) in the appropriate blanks, all 50 words from the list will fit into the matrix. As you find each word and insert the correct vowels, circle the word in the matrix and cross the word off the list. Words may be forward, backwards, up, down, or diagonal, but always in a straight line, never skipping letters. However, some of the letters are used more than once. After circling all the words on the list, the seven remaining letters (including two blanks for vowels) in the matrix will spell the name of a high level computer programming language these words are related to. Be careful, though; some words may appear to fit in more than one place in the matrix. There is, however, only one correct position for each word, so that all the words from the list will be used.

ARRAY
ASSIGN
BUG
CALL
COMPLEX
*DATA
*DECODE
DIMENSION
ENCODE
*END
ENTRY
EQUIVALENCE

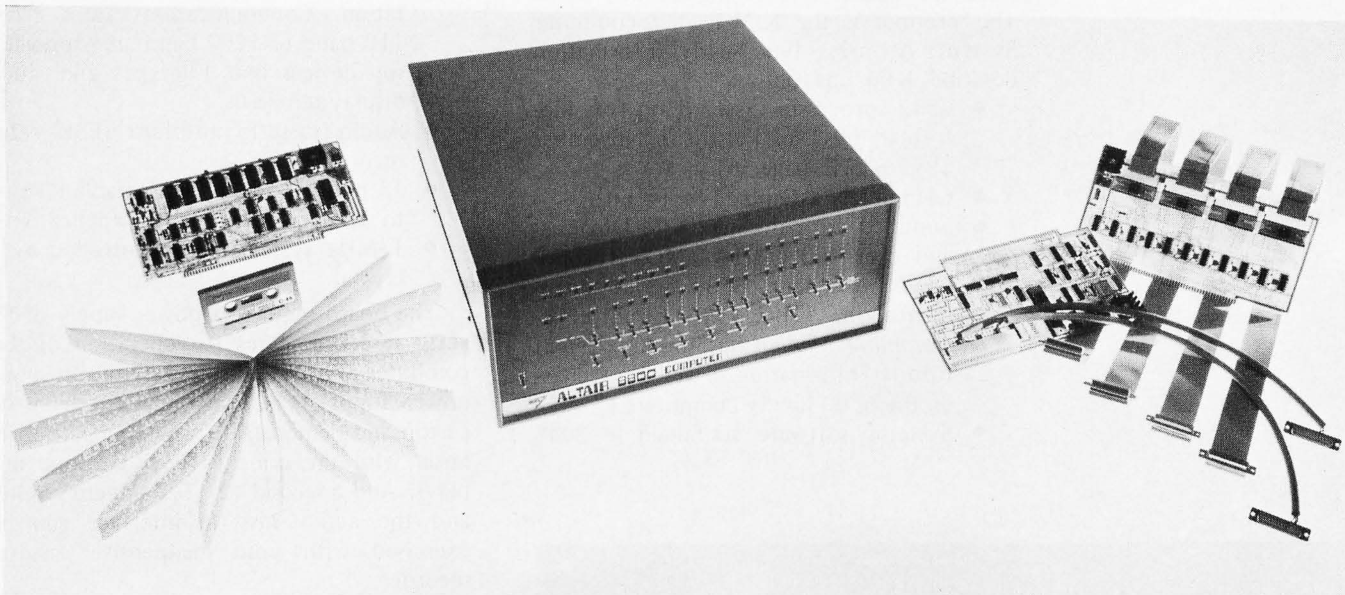
EXPRESSION
EXTERNAL
FIELD
FIND
FUNCTION
*GOTO
IMPLICIT
INTEGER
*LABEL
LIBRARY
LITERAL
LOGARITHM

LOGICAL
LOOP
NAMELIST
*NOT
OCTAL
OPERATOR
OUTPUT
PAUSE
PRECISION
PRINT
PROGRAM
PUNCH
*READ

*REAL
RECORD
RETURN
REWIND
SCALAR
SPACE
STOP
SUBROUTINE
*TAG
TYPE
*UNIT
VARIABLE
WRITE

*** = words that appear to fit in more than one location, but only one position is correct.**

For a limited time only, you can own an Altair® 8800 Computer kit with 4,096 words of memory, new Altair multi-port interface, and revolutionary Altair BASIC language software, for just $695. A savings of up to $114!*



Computer. The Altair 8800 is the best-selling general-purpose computer in the world today. It is a parallel 8-bit word/16-bit address computer with an instruction cycle time of 2 microseconds. It was designed for almost unlimited peripheral and memory expansion, using a bus system where all input/output connections merge into a common line. The Altair 8800 is capable of addressing up to 65,536 words (bytes) of memory. *Regularly priced at $439 for a kit and $621 assembled.*

Memory. The Altair 4K Memory Board provides 4,096 words of dynamic random-access-memory for the Altair 8800. Contains memory protect circuitry, and address selection circuitry for any one of 16 starting address locations in increments of 4K. Access time is 200-300 nanoseconds. The entire 4,096 words of memory on the board can be protected by switching to PROTECT. *Regularly priced at $195 for kit and $275 assembled.*

Interface. Your choice—either the *new Altair 88-2SIO serial interface* or the *new Altair 88-4PIO parallel interface*. The serial interface can be ordered with either one or two ports and the parallel interface can be ordered with up to four ports. *Add $24 for an additional 88-2SIO port kit. Add $30 for each additional 88-4PIO port kit.*

Each port of the *new serial interface board* is user-selectable for RS232, TTL, or 20 milliamp current loop (Teletype). The 88-2SIO with two ports can interface two serial I/O devices, each running at a different baud rate and each using a different electrical interconnect. For example, the 88-2SIO could be interfaced to an RS232 CRT terminal running at 9600 baud and a Teletype running at 110 baud. An on-board, crystal-controlled clock allows each port to be set for one of 12 baud rates. *The 88-2SIO is regularly priced at $115 kit and $144 assembled.*

Each port of the *new parallel interface board* provides 16 data lines and four controllable interrupt lines. Each of the data lines can be used as an input or output so that a single port can interface a terminal requiring 8 lines in and 8 lines out. All data lines are TTL compatible. *The 88-4PIO regularly sells for $86 kit and $112 assembled.*

Software. Altair 4K BASIC leaves approximately 725 bytes in a 4K Altair for programming which can be increased by deleting the math functions (SIN, SQR, RND). This powerful BASIC has 16 statements (IF . . . THEN, GOTO, GOSUB, RETURN, FOR, NEXT, READ, INPUT, END, DATA, LET, DIM, REM, RESTORE, PRINT, and STOP) in addition to 4 commands (LIST, RUN, CLEAR, NEW) and 6 functions (RND, SQR, SIN, ABS, INT, TAB, and SGN). Other features include: direct execution of any statement except INPUT: an "@" symbol that deletes a whole line and a "←" that deletes the last character; two-character error code and line number printed when error occurs; Control C which is used to interrupt a program; maximum line number of 65,529; and all results calculated to seven decimal digits of precision. *Altair 4K BASIC is regularly priced at $60 for purchasers of an Altair 8800, 4K of Altair memory, and an Altair I/O board. Please specify paper tape or cassette tape when ordering.*

*** Savings depends upon which interface board you choose. An Altair 4K BASIC language system kit with an 88-2SIO interface regularly sells for $809. With an 88-4PIO interface, this system sells for $780.**

NOTE: Offer expires on March 30, 1976.

MITS

"Creative Electronics"

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

MAIL THIS COUPON TODAY!

☐ Enclosed is check for $________
☐ BankAmericard #________ ☐ or Master Charge #________
☐ Altair BASIC System Special ☐ 4PIO interface ☐ 2SIO interface
☐ Cassette tape ☐ or paper tape
☐ Extra SIO port ☐ ____ Extra 4PIO ports
Add $8 for postage and handling.
☐ Please send free literature
NAME________
ADDRESS________
CITY________ STATE & ZIP________

NOTE: Personal checks take 2-3 weeks for clearance. For immediate processing send money order or use charge card. Delivery: 30 days. Prices, specifications and delivery subject to change.

What's New?

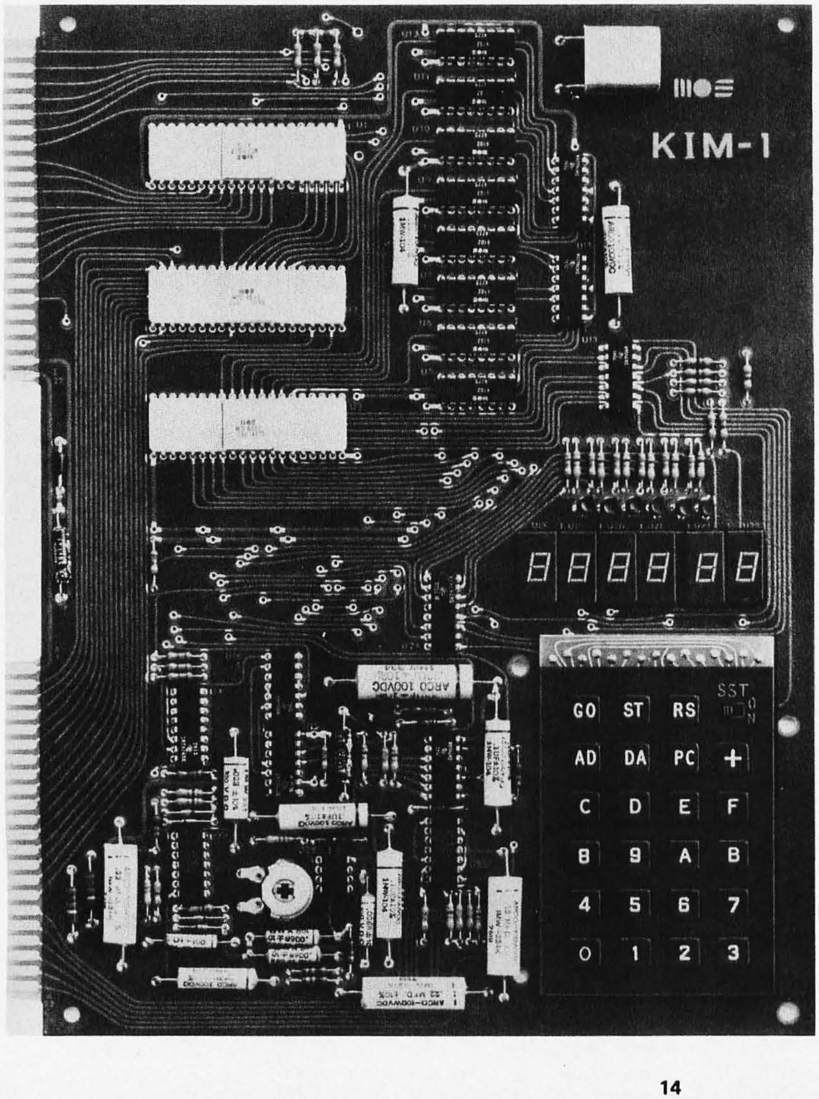
What's New, KIM-o-sabee?

MOS Technology Inc has announced a new microcomputer system which is being marketed to individual hobbyists as well as to the standard industrial markets. This marks a "first" for the personal systems marketplace — a semiconductor manufacturer recognizing the potential of the hobbyist market and selling directly to it. The product is the KIM-1 Microcomputer System. At press time, early information describes KIM-1 as follows:

- 6502 processor (see "Son of Motorola" by Dan Fylstra in November 1975 BYTE, page 56).
- Completely assembled (not a kit).
- Supplied with the new KIM-1 manual and over 400 pages of MOS Technology's excellent 6500 series documentation. (The *6500 Family Programming Manual* has some excellent tutorial information as well as specifics on the 6500 family computers.)
- Systems software contained in 2048 bytes of ROM in two 6530 ROM/RAM/IO arrays.
- 1024 bytes of static user RAM.
- 23 key keyboard for programmed inputs and control of the monitor.
- 6 digit LED display for programmed outputs and monitor displays.
- General purpose serial interface with automatic line speed sensing and adaptation. Communications rates from 110 baud to 1200 baud are supported for devices like Teletypes and video display terminals.
- Audio cassette interface (FSK ratio recording).
- 15 bidirectional programmable IO pins to control experimental applications.
- 1 MHz system clock controlled by a crystal.

The board requires a power supply of +5 volts at 1.2 amperes for operation of the computer and LED displays. With this single power supply, you can unpack it from its carton and demonstrate programmable operation with the onboard keyboard and displays. Add a second 12V 0.1 ampere supply, and the audio cassette interface can be exercised with your inexpensive cassette recorder.

This product will prove attractive to readers who are not inclined to fondle hardware extensively, but want a programmable machine with the minimum amount of trouble.



Glorobots

℞

A robot was having conniptions
at reading handwritten inscriptions,
but acquired the knack
by decoding a stack
of typical doctor prescriptions.

Evolution

A self-evolved robot named Babbitt,
because of his dubious habit
of unbridled mating
and self-propagating,
was housed in a hutch like a rabbit.

Hear Ye Hear Ye

The sensory robots are near,
but will not be ready this year,
for each of them tries
to eat with his eyes,
and cocks his nose trying to hear.

Gloria Maxson
13602 Cullen
Whittier CA 90605

Programming the Implementation

Look for logical gaps in the design of a specific system, then ask yourself: "What would happen if I did something that is technically undefined for my computer?"

Charles A Crayne
734 S Ardmore Av
Los Angeles CA 90005

An exploration of the Scelbi 8H reveals two single byte instructions which are artifacts of the implementation: Load minus 1 immediate (L1I) and read option switches (ROS).

When Scelbi Computer Consulting Inc designed their 8H minicomputer system around the Intel 8008 CPU chip, they implemented two instructions beyond those available on the Intel chip. It seems that these two instructions are a byproduct of the 8H design, and of potential interest to the hobbyist.

The term "architecture" generally means the design of a system, or of a family of systems. The instruction set, maximum address, number of ports, etc., are architectural considerations affecting the design of a chip. The term "implementation" refers to the technical methods used to bring the defined capabilities into existence; this may be accomplished by backplane wiring, or ROM, number and speed of clocks, choice of power supplies, etc.

The distinction between architecture and implementation is clearer for a computer "family" than for a single device. In the IBM line, for example, all 370s share a single architecture. The difference between a model 115 and a model 168 is implementation. Note also that a designer is not required to implement the full capacity provided by the architecture. On the Scelbi 8H, the architecture provides for 16 K bytes of main storage, but the implementation allows for only 4 K bytes.

At first glance, the matter of the Scelbi 8H input ports is just another example of implementing less than the full architectural capacity. Eight input ports are allowed for in the instruction set, but only six are provided. A surprise is in store, however, for anyone who asks himself the question, "What would the computer do if I called for input from port 6 or port 7?" (The eight possible ports are numbered from 0 to 7.)

To answer this question, it is necessary to consider just why Scelbi decided to provide only six input ports. This can be resolved by a glance at the logic diagram for the Scelbi input board. The function of this board is to couple the 8008 data bus, at the appropriate times, to one of the input ports, the external memory data bus, or (in the case of a front panel interrupt) to the front panel data switches. This switching operation is implemented with type 74151 ICs, which are one of eight data selectors. That is, they will accept eight bits of input, and will output one of the eight, depending upon the three bit address (000 to 111) supplied at the time the enable line goes high.

The Scelbi designer needed a one of ten selector to accommodate the eight input ports, the memory bus, and the panel switches. But binary addressing just doesn't work that way. He could use a one of 16 selector and leave six inputs unused, or he could do as he actually did and use the 74151s by reducing the number of input ports implemented to six. This decision resulted in a lower cost system. Logic is provided to force the selector addresses to 110 for memory input operations, and to 111 during interrupt processing. But, again, probably for cost reasons no logic is provided to assure that the address specified in an input instruction is limited to the 000 to 101 range (IN0 to IN5).

Therefore, the IN6 instruction (input from port 6) causes the accumulator to be set from the memory bus, and the IN7 instruction causes the accumulator to be set from the front panel switches. Unlike the LAM instruction, however, IN6 does not cause the memory address to be developed. Therefore, the result of the instruction is to load the accumulator with the value −1 or 377 in octal. This is not exactly a big deal, as it is exactly the same effect as would result from requesting an input from any unconnected port. Still, it can be of some use in saving memory space in exactly the same way as the experienced programmer uses an

BIG HELP FOR SMALL BUCKS.

Used to be, filling your bench with really good test equipment meant emptying your wallet. Because you had to pay a price for quality and versatility. A high price.

Not anymore. CSC's ingenious new Design Mate™ line of test equipment combines performance high enough for most lab applications with the kind of versatility and economy everyone can afford. Thanks to a number of new design concepts and manufacturing techniques.

Proof? Check the specs and prices below, then visit your CSC dealer for a demonstration. Or write for our catalog and distributor list.

Design Mate 1 Breadboard/ Power Supply. **$49.95***



Design Mate 2 Wide-Range Function Generator. **$64.95***

Design Mate 3 Precision R/C Bridge. **$54.95***

CONTINENTAL SPECIALTIES CORPORATION

CSC

EASY DOES IT

44 Kendall Street, Box 1942
New Haven, CT 06509 • 203-624-3103 TWX: 710-465-1227
West Coast office: Box 7809, San Francisco, CA
94119 • 415-421-8872 TWX: 910-372-7992
Canada: Len Finkler Ltd., Ontario

*Manufacturer's suggested list
Prices and specifications subject to change without notice.

XRA instruction instead of a LAI 000 instruction to clear the accumulator. This "load −1" instruction only occupies one byte of program space as compared to the two bytes required by an ordinary immediate instruction.

The IN7 instruction is more useful. The ability to dynamically interrogate the front panel switches during the execution of the program allows the operator to modify the program operation without the need for complex console communication routines. This capability was known as program option switches on the IBM 1401 and similar machines and was once widely used in commercial applications.

Suppose that one had a program which printed mailing labels for a computer newsletter. To save postage, it is desired to send a certain issue only to those who have indicated an interest in the specific topics covered. For example, some persons want to read only about hardware design, while others are interested in applications, compiler writing, etc. Multiple interests are allowed.

If up to eight areas of interest are coded as bits in an option byte included in the machine readable address list, then a program can create any desired subset of the list just by comparing to see if any bit in the option byte matches the corresponding bit in the program option switches. (The programming to do this is quite simple. Point the memory registers to the option byte in the file buffer, read the switches with IN7, and the option byte with the NDM instruction, and skip printing with a JTZ if the option does not match the switches.)

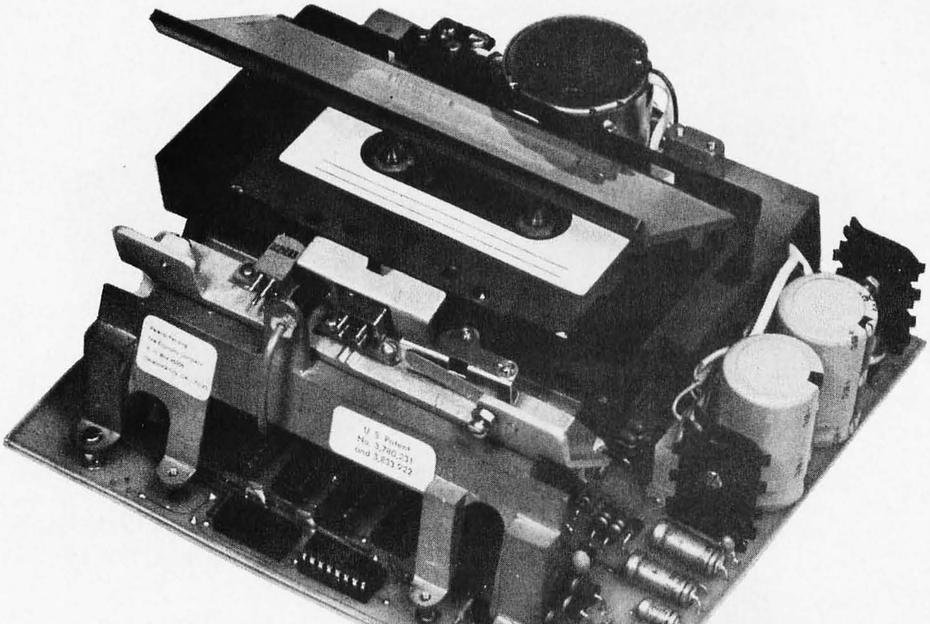
Another possible application would be to write an output subroutine such that it checks the program option switches to see if it should write to a video display, a hardcopy device, or both. This technique could save a lot of paper during program checkout, or when just demonstrating an application, and still allow permanent reports when required.

If you don't have a Scelbi, you can still implement program option switches yourself by putting a few switches in a minibox and hooking it up as a regular input device. But the interesting question is: What hidden capabilities does your system have? All you need is a little imagination and an inquiring mind. Look at your own system. Compare the users manual with the logical structure in the CPU chip manufacturer's catalogue. Look for logical inconsistencies and restrictions. Try some experiments with your equipment. Maybe you too can discover some new abilities for your system. ■

Whatever your computer, get to know the implementation as well as possible. You may be able to find similar hidden instructions.

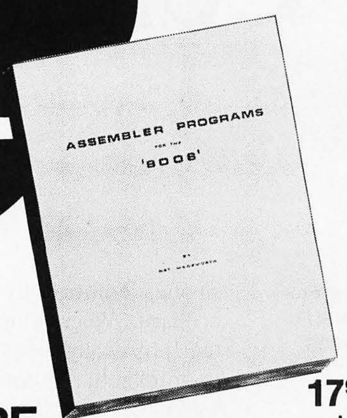
Architecture refers to overall system design — implementation refers to the specific technical methods.

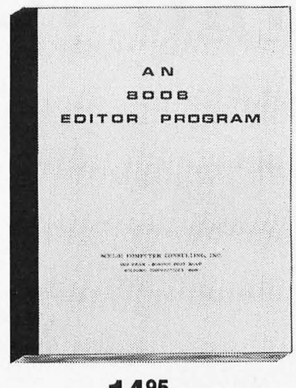
What's New?

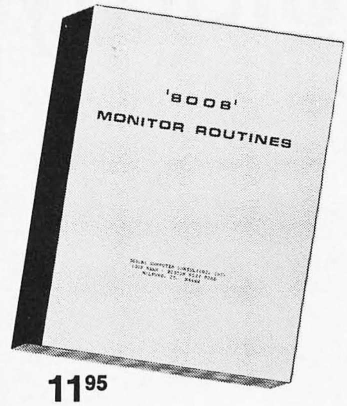


Here is a product which will be of interest to many BYTE readers. This is an electronically controlled variable speed digital cassette deck which can be adapted for use by the home computer experimenter. It is made by Triple I, a division of the Economy Company, PO Box 25308, Oklahoma City OK 73125. Its price is in the $100 range and it should prove to be an excellent medium for totally automated data storage in personal computing systems.

SCELBI SOFTWARE
FOR BEGINNERS







"8008" SOFTWARE MANUALS

17⁹⁵ ppd.

Assembler Programs For The "8008"
Minimum length assembler program for 2K memory, plus sophisticated version for additional memory or more power. Includes fundamentals of operation, how to format "source listing", step-by-step analysis/presentation of subroutines, flow charts, listings of programs and lots more. Order yours today!

14⁹⁵ ppd.

"8008" Editor Program
Describes variations that can reside in 2K of memory. How to increase buffer size. How to manipulate "text" for "source listings" and other text preparations, routines discussions, flow charts, assembled listings, more. Order now!

11⁹⁵ ppd.

"8008" Monitor Routines
Describes "Monitor Control" package to control operation from external "keyboard". Routines to examine and modify memory locations, CPU registers, set "breakpoints", "debugging" programs, control bulk storage I/O devices, and more. Includes subroutine explanations, flow charts, assembled highly commented program listing. Order today!

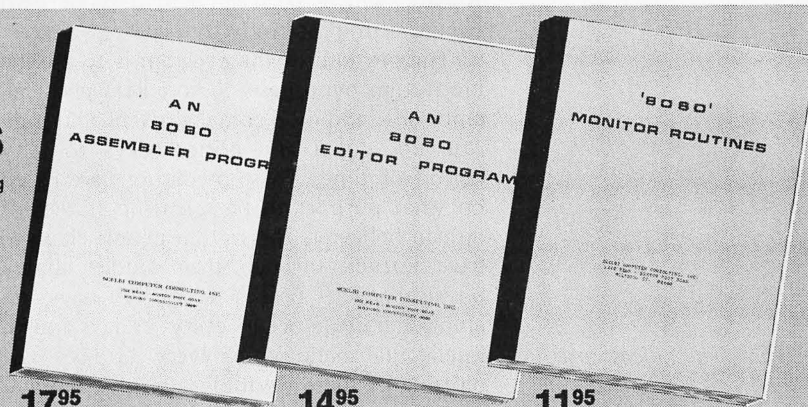
SAVE $5.35! BUY ALL 3 FOR ONLY $39.50!

"8080" SOFTWARE MANUALS

Three new basic, complete "MUST" manuals! **"8080" Assembler Program** operates easily in 4K bytes of RAM (including symbol table). Unique feature: How to accept mnemonics related to "8008" based machines on "8080". Includes all major routines, pertinent flow charts, highly commented assembled listing and more. **$17.95.**

"8080" Editor Program identical to "8008" Editor, with machine codes for "8080". **$14.95.**

"8080" Monitor Routines same functions as "8008" specifically developed to utilize expanded capabilities of "8080". **$11.95.**



17⁹⁵ ppd. **14⁹⁵ ppd.** **11⁹⁵ ppd.**

SAVE $5.35! BUY ALL 3 FOR ONLY $39.50!



19⁹⁵ ppd.

MACHINE LANGUAGE PROGRAMMING FOR THE "8008"
(and similar microcomputers)

- Detailed presentation of "8008" codes • Flow Charts
- Mapping • Floating-point Package • Debugging
- Basic programs: loops, counters, masks
- Organizing Tables • Editing/Assembling
- Math operations • I/O, Real Time Programming
- Maximizing memories • And lots more.

Here's the detailed, basic manual you need to develop today's machine language programs. 170 pages. Illustrated. Easy-to-read, understand. Most techniques applicable to other micro-computers, including "8080". Floating-point arithmetic package is worth the price alone!

Order today!

Pricing, specifications, availability subject to change without notice. Prices for U.S. and Canadian delivery at book mailing rate. Add $2.50 for each publication if Priority Air Service (U.S.) desired. Overseas include $5.00 for each publication for Airmail Service.

SCELBI COMPUTER CONSULTING INC.
1322 Rear Boston Post Rd., Milford, CT 06460
Telephone: 203/874-1573

Biorhythm for Computers

Joy and Richard Fox
1364 Campbell St
Orlando FL 32806

According to the biorhythm hypothesis, there is a reason for those doldrum days when even your computer refuses to communicate with you.

[NOTE: *The ideas presented in this article are a hypothesis about human mental states and are not necessarily a valid predictive theory. One danger of computer programming is the assumption that a logically correct program which executes without bombing out will necessarily produce meaningful results. Whatever the final conclusion with regard to the biorhythm hypothesis, the calculation makes an interesting example of a BASIC language application program. . . .* CH]

There is no doubt that all living things have biological rhythms. The study of three of these rhythms in humans has led to the development of a pseudo science, biorhythm, that, through the use of computers, is growing in popularity in the United States. This article describes a program, written in BASIC, which you can run in your own computer to plot biorhythm curves.

The purpose of the program is to use the biorhythm hypothesis to "predict" physical, emotional and intellectual patterns that indicate up, down and critical days for any period of time. These predictions are based on what purport to be scientific studies of human behavior. Biorhythm people claim to have learned through their studies that a physical cycle occurs every 23 days, an emotional cycle occurs every 28 days and an intellectual cycle occurs every 33 days. The plotting of these rhythms is printed out as a two-dimensional graph on a Teletype or similar output device, showing the three cycles as a function of time.

The biorhythm hypothesis is nothing new. It was first proposed in the late nineteenth century by a Viennese psychologist and a German physician, each working separately. In the 1920s, an Austrian teacher added the 33 day intellectual cycle after studying the performance of high school and college students.

According to the biorhythm hypothesis, there is a reason for those doldrum days when even your computer refuses to communicate with you. Each of the three cycles oscillates between ups and downs. When your cycles are up, you feel physically strong, emotionally high or intellectually brilliant. When your cycles are down, you feel physically weak, emotionally depressed or intellectually dull. But the days to really watch out for are the transition days when you are crossing from a low to a high or a high to a low. It is during these transition days that you are especially susceptible to accident and illness. A few times each year, two or even all three of your cycles will cross the transition simultaneously. According to biorhythm people, these critical days are best spent quietly.

The biorhythm hypothesis has gained acceptance in a growing number of industries. In Japan, 2,000 businesses use biorhythm calculations. One Japanese firm reports a 35% reduction in computer data errors by assigning workers to other tasks when they are going through critical days. Another Japanese firm using biorhythm predictions claims to have reduced its yearly vehicle accident loss by 45%. An American survey of 1,000 industrial accidents showed that 90% of them occurred on critical days.

Mike Bertalot, a supervisor for United Airlines, estimates that between 6,000 and 8,000 of United's 40,000 employees are using biorhythm predictions as a guide for safety awareness. United uses the printouts, which they distribute to interested employees, as "an excuse to warn employees about safety." The result has been that some departments have shown a 50% decrease in accidents. It is not clear whether this reduction is due to the extra warnings or to the predictive value of the hypothesis. Although the future of the biorhythm experiment at United is uncertain, the results are being sent to the United States Naval Laboratory, which is studying the hypothesis.

Biorhythms have also been used for profit. The September 15 1975 issue of *Newsweek* quotes Lester Cherubin, president

of Time Pattern Research, Inc, as having sold 100,000 biorhythm printouts for $10 to $20 each in the past three years. Other companies sell plastic biorhythm calculating devices for anywhere between $4 and $20. Some shopping center vendors sell for a mere 50 cents a computer printout of your rhythms for one day.

The calculation of biorhythm curves is not easy to do with a pencil and paper. First, the subject's age in days must be calculated. This problem, of course, is complicated by all the peculiarities of the modern calendar. Then you must calculate how many complete 23 day cycles the subject has lived through and how many days he is into the next cycle. (The biorhythm hypothesis makes a simplifying assumption that all cycles originate at birth with zero relative phase.) The same must be done for the 28 and 33 day cycles. The fraction of each cycle is multiplied by two pi radians and the sine of that number is taken to obtain the points of the biorhythm curve for that day. The calculation must be rerun for each succeeding day, and the results plotted on graph paper, in order to obtain the biorhythm curves.

The program to calculate biorhythm curves is shown in the form of a flow chart in figure 1; figure 2 shows the complete listing of this program in BASIC. The operation of the program is as follows:

Line 0001 dimensions the strings N and S and the array T. N will be filled with the character set for the days of the month and S will be filled with the image of each line of the graph, as it is prepared for printing. T will be filled from the data statement at line 0080 with the number of days in each month of the year. The input statement at line 0008 and the if statement at line 0009 together allow the user to skip over the explanatory printout at the beginning of the program and go directly to the calculation which starts at line 0027.

Line 0040 defines the numeric values for the month, day and year that the subject was born. Line 0050 defines the month, day and year for the start of the printout. The year can be supplied as a two digit number ('76) or as a four digit number (1976), but the same format must be used for both the birth date and the printout target date. Line 0065 defines the number of days to be plotted.

D3 in the program is the variable which will contain the age of the subject in days. At line 0130, D3 is initialized to 0. The program will now calculate the number of days between the subject's birth date and the requested plotting date. The calculation is performed in several steps, and at the end of each step, the value calculated at that step is added to the total in D3.

Next, the program checks if the subject was born in January or February of a leap year. The test for a leap year, at line 0150, is made by dividing the birth year by four and checking for a remainder. Only leap years divide by four with a remainder of zero. If the subject was born in January or February of a leap year, one day is added to the running total, at line 0160. Otherwise, the running total is left at zero.

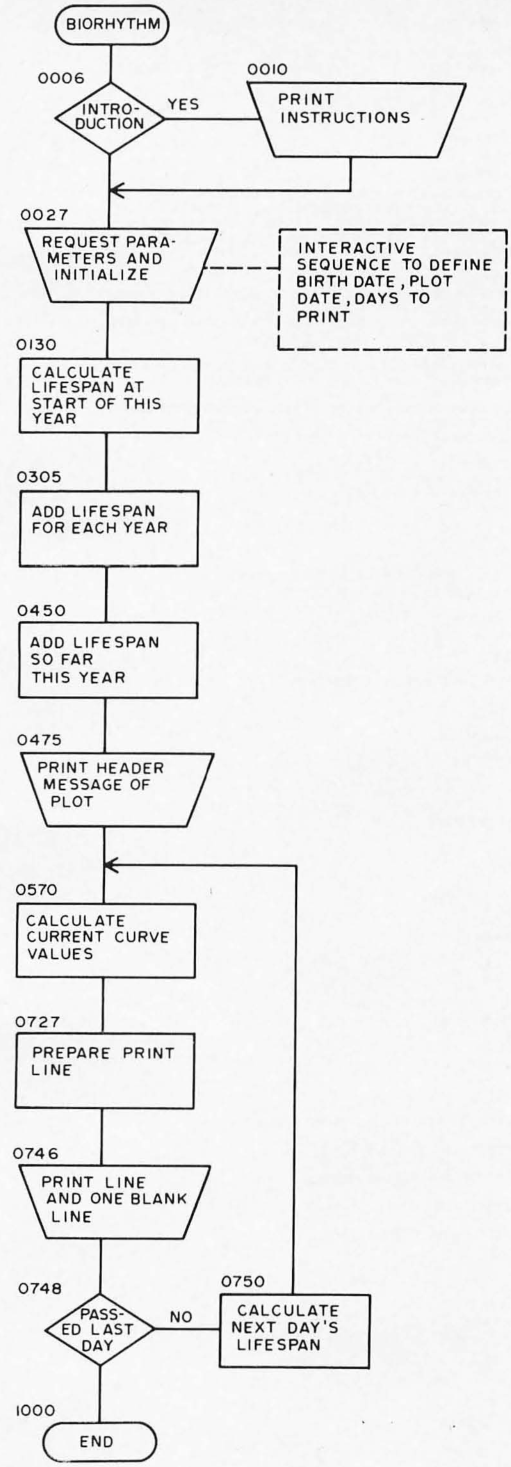


Figure 1: Flow Chart of Biorhythm Calculator. This chart illustrates the general outline of the program found in figure 2. The numbers noted next to symbols in the flow chart refer to line numbers of the listing in figure 2.

Watch out for evil omens on transition days.

While not intended to apply to machines, maybe biorhythms can be used to predict computer behavior. Enter the birth date of your computer and predict when your cybernetic monster plans its next bomb out!

```
0001   DIM NS(72), SS(72), T(12)
0002   NS="0001020304050607080910111213141516171819202122232425262728293031"
0004   REM BIORHYTHM CREATED BY JOY AND RICHARD FOX
0006   PRINT "DO YOU WISH AN INTRODUCTION TO BIORHYTHM? TYPE 1 FOR YES,".
0007   PRINT "OR 0 FOR NO."
0008   INPUT A
0009   IF A=0 THEN 27
0010   PRINT TAB(25), "BIORHYTHM"
0011   PRINT
0012   PRINT
0013   PRINT
0015   PRINT "THE PURPOSE OF BIORHYTHM IS TO PREDICT A PHYSICAL,"
0016   PRINT "EMOTIONAL AND INTELLECTUAL PATTERN THAT INDICATES YOUR"
0017   PRINT "UP AND DOWN DAYS FOR ANY PERIOD OF TIME. BIORHYTHM CAN"
0018   PRINT "SHOW WHICH DAYS WERE GOOD OR BAD FOR YOU BEGINNING WITH"
0019   PRINT "YOUR BIRTH. IT CAN ALSO SHOW YOU WHICH FUTURE"
0020   PRINT "DAYS WILL BE GOOD OR BAD FOR YOU."
0021   PRINT "THESE PREDICTIONS ARE BASED ON SCIENTIFIC"
0022   PRINT "STUDIES TO DETERMINE WHY ACCIDENTS OCCUR. IT WAS LEARNED"
0023   PRINT "THROUGH THESE STUDIES THAT A PHYSICAL CYCLE OCCURS EVERY"
0024   PRINT "23 DAYS, AN EMOTIONAL CYCLE OCCURS EVERY 28 DAYS, AND AN"
0025   PRINT "INTELLECTUAL CYCLE OCCURS EVERY 33 DAYS."
0026   PRINT
0027   PRINT "PLEASE TYPE YOUR BIRTH DATE USING THE FOLLOWING FORMAT "
0028   PRINT "MM,DD,YY. EXAMPLE: JANUARY 17, 1942 = 01,17,42"
0040   INPUT M, D, Y
0045   PRINT "AT WHAT DATE ARE YOU INTERESTED IN BEGINNING BIORHYTHM?"
0050   INPUT M1, D1, Y1
0060   PRINT "HOW MANY DAYS DO YOU WISH TO HAVE PLOTTED?"
0065   INPUT D2
0080   DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
0110   REM M=MONTH, D=DAY, Y=YEAR
0120   REM D3=TOTAL NUMBER OF DAYS ELAPSED
0130   D3=0
0140   IF M>2 THEN 200
0150   IF INT(Y/4)-(Y/4)<>0 THEN 200
0160   D3=1
0200   FOR I=1 TO 12
0210   READ T(I)
0220   REM T=DAYS IN EACH MONTH
0230   NEXT I
0240   D3=T(M)-D+D3
0250   FOR I=M+1 TO 12
0260   D3=T(I)+D3
0270   NEXT I
0280   REM Y3=YEAR COUNTER FROM BIRTH TO DISPLAY
0290   Y3=Y
0299   Y3=Y3+1
0305   IF Y3>=Y1 THEN 400
0310   IF INT(Y3/4)-(Y3/4)=0 THEN 320
0315   D3=D3+365
0316   GOTO 299
0320   D3=D3+366
0325   GOTO 299
0400   IF M1<=2 THEN 450
0405   IF INT(Y1/4)-(Y1/4)<>0 THEN 450
0410   D3=D3+1
0450   FOR I=1 TO M1-1
0455   D3=T(I)+D3
0460   NEXT I
0470   D3=D1+D3
0475   PRINT "PHYSICAL CYCLE = P"
0480   PRINT "EMOTIONAL CYCLE = E"
0490   PRINT "INTELLECTUAL CYCLE = I"
0491   PRINT
0492   PRINT
0493   PRINT
0500   PRINT "DATE";
0505   PRINT TAB(13), "DOWN";
0510   PRINT TAB(35), "CRITICAL";
0520   PRINT TAB(63), "UP"
0525   PRINT "......            ..........................................";
0526   PRINT "................"
0530   LET M4=M1
0540   LET D4=D1
0550   LET Y4=Y1
0560   REM M4,D4,Y4 = DATE PRINTED OUT IN PLOTTING CHART
0570   GOTO 580
0571   REM F=FRACTION INTO CYCLE
0580   F=(D3/23)-INT(D3/23)
0600   REM X=THE ARGUMENT FOR THE SINE FUNCTION
0610   X=F*2*3.1416
0640   REM P=THE PHYSICAL POSITION ON THE GRAPH
0650   P=((SIN(X)+1)*24)+15
0655   REM E=EMOTIONAL POSITION ON THE GRAPH
0660   F=(D3/28)-INT(D3/28)
0670   X=F*2*3.1416
0680   E=((SIN(X)+1)*24)+15
0690   F=(D3/33)-INT(D3/33)
0700   X=F*2*3.1416
0710   REM I=INTELLECTUAL POSITION ON THE GRAPH
0720   I=((SIN(X)+1)*24)+15
0727   FOR X=1 TO 32
0728   S$(2*X-1,2*X)="  "
0729   NEXT X
0731   S$(39,39)=" "
0732   S$(P,P)="P"
0733   S$(E,E)="E"
0734   S$(I,I)="I"
0735   S$(3,3)="/"
0736   S$(6,6)="/"
0741   S$(1,2)=N$(M4*2+1,M4*2+2)
0742   S$(4,5)=N$(D4*2+1,D4*2+2)
0743   IF Y4>99 THEN 950
0744   S$(7,7)=N$((INT(Y4/10)*2+2),(INT(Y4/10)*2+2))
0745   S$(8,8)=N$((Y4-INT(Y4/10)*10)*2+2,(Y4-INT(Y4/10)*10)*2+2)
0746   PRINT S$(1,63)
0747   PRINT
0748   IF D2=1 THEN 1000
0750   D2=D2-1
0800   D3=D3+1
0810   D4=D4+1
0815   IF M4<>2 THEN 820
0816   IF D4<>29 THEN 820
0817   IF INT(Y4/4)-(Y4/4)<>0 THEN 820
0818   GOTO 570
0820   IF D4<=T(M4) THEN 570
0830   M4=M4+1
0835   D4=1
0840   IF M4>12 THEN 870
0850   GOTO 570
0870   M4=1
0880   Y4=Y4+1
0900   GOTO 570
0950   Y4=Y4-(INT(Y4/100)*100)
0951   GOTO 744
1000   END
```

Figure 2: BASIC Program of the Biorhythm Calculator. This is the complete listing of a BASIC program to perform calculations and plot the results on a hard copy printer.

Lines 0200 through 0230 fill the array T with the values in data statement 0080 so that the array contains the number of days in each month of the year. Line 0240 calculates the number of days from the subject's birth date to the end of his first calendar month, and adds that number to the running total in D3. Lines 0250 through 0270 calculate the number of days in each month during the remainder of the subject's birth year, and add that number to the running total.

The birth year, Y, is transferred to the year counter Y3, and the year counter is incremented at line 0299. If the year counter is greater than or equal to the year to be printed out, Y1, then the program jumps to line 0400. Otherwise, the program adds 365 or 366 to the running total for each year between birth and the target year. Each time that is done, the year counter is incremented. When it matches the printout target year, the program jumps to line 0400.

Next the program calculates the number of days between the start of the display year and the display day. If the display month is March or later, then the program checks if the display year is a leap year. If it is, one day is added to the running total at line 0410. Lines 0450 through 0460 add the number of days in each month between the start of the display year and the display month to the running total D3. Line 0470 adds the number of days into the display month to the running total. D3 now contains the age of the subject in days, as of the requested display date.

Lines 0475 through 0526 print the header of the graph. Lines 0530 through 0571 set up three new variables, M4, D4, and Y4, which will contain each consecutive date as it is printed out.

Now the program calculates the phase of each of the three biological cycles for the subject for the dates requested. The physical cycle has a period of 23 days. If you divide the age of the subject in days by 23, the remainder is a number between 0 and 22.9. That remainder is proportional to the phase of the subject's physical cycle at the requested date. The remainder is stored in variable F at statement 0580. F is then multiplied by two pi radians and the answer is stored in X. X is therefore a number between zero and two pi and is proportional to the phase of the subject's physical cycle. Line 0650 takes the sine of X. The result is a value between +1 and −1. This number is then normalized to a value between 15 and 63 and is stored in P. The values 15 and 63 represent the beginning and ending column numbers of the graph on the Teletype.

Extreme down days will plot in column 15. Extreme up days will plot in column 63. Critical days will plot in column 39, and other days will plot in between these points.

The same calculation is then repeated at lines 0660 through 0680, with a period of 28 days, for the emotional cycle; and at lines 0690 through 0720, with a period of 33 days, for the intellectual cycle. Lines 0727 through 0729 loop to fill up the string S with blank characters, to wipe out old data still in the string. Line 0731 places a dot character in element 39 of the string, so that the zero crossing will be clearly marked by a string of dots down the 39th column of the page. Line 0732 stores the character "P" into the column calculated by the equation for the physical cycle. Lines 0733 and 0734 do the same for the characters "E" and "I". Next the program places slashes in elements three and six of the string S, so that they will print out as slashes in the date at the left of the graph.

The month is placed in array elements one and two and the day is placed in elements four and five. If the operator typed the year as a four digit number, the program truncates the most significant two digits. Line 0744 places the ten's digit of the year into element seven of the string and line 0745 puts the unit's digit of the year into element eight.

The string S is now ready for printing. Line 0746 prints elements one through 63 across the output device page as a month, day, year, a dot at column 39 and the letters "P", "E" and "I" in appropriate positions. Line 0747 causes the typewriter to double space so the graph is easier to read.

If the number of days left to print, D2, has been reduced to one, then the program exits. Otherwise, D2 is decremented by one, and the age of the subject in days is incremented by one.

The date in the month, D4, is incremented and the program checks if the day to be plotted is February 29 of leap year. If it is, the next day's data is plotted. If it is not February 29 of a leap year, then the number of days in the month is checked against the maximum number of days in that month as defined in table T. If the day in the month, D4, is too large, it is reset to one and the month is incremented. If the month has been incremented to 13, it is reset to one and the year is incremented. The program prints the next day's data and keeps looping till all the requested data has been printed.

This program has an unusual application that you may not yet have considered: enter the birth date of your computer, and predict when your cybernetic monster plans its next bomb out!!!■

```
DO YOU WISH AN INTRODUCTION TO BIORHYTHM? TYPE 1 FOR YES,OR 0 FOR NO.
?1
                              BIORHYTHM

THE PURPOSE OF BIORHYTHM IS TO PREDICT A PHYSICAL,
EMOTIONAL AND INTELLECTUAL PATTERN THAT INDICATES YOUR
UP AND DOWN DAYS FOR ANY PERIOD OF TIME.  BIORHYTHM CAN
SHOW WHICH DAYS WERE GOOD OR BAD FOR YOU BEGINNING WITH
YOUR BIRTH. IT CAN ALSO SHOW YOU WHICH FUTURE
DAYS WILL BE GOOD OR BAD FOR YOU.
THESE PREDICTIONS ARE BASED ON SCIENTIFIC
STUDIES TO DETERMINE WHY ACCIDENTS OCCUR.  IT WAS LEARNED
THROUGH THESE STUDIES THAT A PHYSICAL CYCLE OCCURS EVERY
23 DAYS, AN EMOTIONAL CYCLE OCCURS EVERY 28 DAYS, AND AN
INTELLECTUAL CYCLE OCCURS EVERY 33 DAYS.

PLEASE TYPE YOUR BIRTH DATE USING THE FOLLOWING FORMAT:
MM,DD,YY. EXAMPLE: JANUARY 17, 1942 = 01,17,42
?01,17,42
AT WHAT DATE ARE YOU INTERESTED IN BEGINNING BIORHYTHM?
?11,25,75
HOW MANY DAYS DO YOU WISH TO HAVE PLOTTED?
?40
PHYSICAL CYCLE = P
EMOTIONAL CYCLE = E
INTELLECTUAL CYCLE = I

DATE        DOWN                  CRITICAL                    UP
------     ---------------------------------------------------------
11/25/75       I      PE              .
11/26/75      I   PE                  .
11/27/75      IPE                     .
11/28/75      I                       .
11/29/75      E I                     .
11/30/75      E P I                   .
12/01/75        E   PI                .
12/02/75           E     I            .
12/03/75               E     I P      .
12/04/75                   E     I  P
12/05/75                        E     I     P
12/06/75                              E   I       P
12/07/75                              .    E  I         P
12/08/75                              .        E I        P
12/09/75                              .            E I       P
12/10/75                              .                EI    P
12/11/75                              .                     IP
12/12/75                              .                  P    I
12/13/75                              .               P       IE
12/14/75                              .         P             I
12/15/75                              .  P                  EI
12/16/75                          P   .                 E  I
12/17/75                    P         .             E    I
12/18/75              P               .         E   I
12/19/75          P                   .     E    I
12/20/75       P                     E.      I
12/21/75      P                  E    . I
12/22/75      P             E       I .
12/23/75         P     E       I      .
12/24/75            EP      I         .
12/25/75         E     I  P           .
12/26/75      E    I            P     .
12/27/75      E   I                   P
12/28/75      EI                      .      P
12/29/75      I E                     .            P
12/30/75      I    E                  .                 P
12/31/75      I        E              .                     P
01/01/76         I          E         .                       P
01/02/76           I             E    .                       P
01/03/76              I               E                      P
```

Figure 3: Output of the Biorhythm Calculator. Here is a listing of the output of the program found in figure 2. In this case, the introductory text was printed prior to entering the parameter definition sequence.

The Magic of Computer Languages

Theodor Nelson
25 S Seventh Av
LaGrange IL 60825

A computer language is a system for casting spells. This is not a metaphor but an exactly true statement. Each language has a vocabulary of *commands*, that is, different orders you can give that are fundamental to the language, and a *syntax*, that is, rules about how to give the commands right, and how you may fit them together and entwine them.

Learning to work with one language doesn't mean you've learned another. You learn them one at a time, but after some experience it gets easier.

There are computer languages for testing rocketships and controlling oil refineries and making pictures. There are computer languages for sociological statistics and designing automobiles. And there are computer languages which will do any of these things, and more, but with more difficulty because they have no purpose built in. (But each of these general purpose languages tends to have its own outlook.)

Most programmers have a favorite language or two, and this is not a rational matter. There are many different computer languages – in fact thousands – but what they all have in common is *acting on series of instructions.* Beyond that, every language is different. So for each language, the questions are

WHAT ARE THE INSTRUCTIONS?
and
HOW DO THEY FIT TOGETHER?

Most computer languages involve somehow typing in the commands of your spell to a computer set up for that language. (The computer is set up by putting in a bigger program, called the *processor* for that language.) Then, after various steps, you get to try your program.

Once you know a language you can cast spells in it; but that doesn't mean it's easy. A spell cast in a computer language will make the computer do what you want –

IF it's possible to do it
  with that computer;
IF it's possible to do it
  in that language;
IF you used the vocabulary
  and rules of the language
  correctly;
and IF you laid out in the spell
  a plan that would effectively
  do what you had in mind.

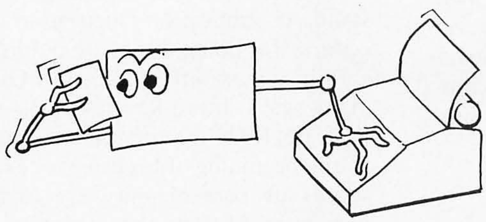
BUT if you make a mistake in casting your spell, that is a BUG. (As you see from the IFs above, many types of bug are possible.) Program bugs can cause unfortunate results. (Supposedly a big NASA rocket failed in takeoff once because of a misplaced dollar sign in a program. [The person responsible for that programming error became a fanatic about bugs and was a key to the success of eyeball debugging of Apollo flight programming . . . CH]) Getting the bugs out of a program is called debugging. It's very hard.

Designing Computer Languages

Every programmer who's designed a language, and created a processor for it, had certain typical uses in mind. If you want to create your own language, you figure out

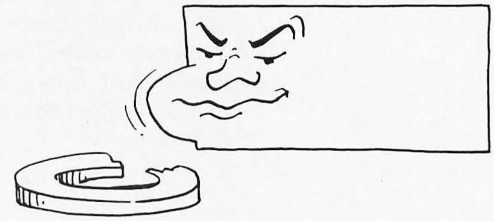
★ AN INTERPRETER ★

carries out each instruction
as it's encountered.



★ A COMPILER ★

chews the instructions
of the language
into another form
to be processed later.



Drawings by Bill Morello, adapted from Nelson's drawings in *Computer Lib/Dream Machines*.

what sorts of operations you would like to have be basic in it, and how you would like it all to fit together so as to allow the variations you have in mind. *Then* you program your processor (which is usually very hard.)

HOW DO COMPUTER LANGUAGES WORK?

Basically there are two different methods.

A *compiling language*, such as FORTRAN or COBOL, has a compiler program, which sits in the computer, and receives the input program, or source program, the way the assembler does. It analyzes the source program and substitutes for it an object program, in machine language, which is a translation of the source program, and can actually be run on the computer. The relation of the higher language is not one to one to machine language: many instructions in machine language are often needed to compile a single instruction of the source program. (A source program of 100 lines can easily come out a thousand lines long in its output version.) Moreover, because of the interdependency of the instructions in the source program, the compiler usually has to check various arrangements all over the program before it can generate the final code.

Most compilers come in several stages. You have to put the *first* stage of the compiler into the computer, then run in the source program, and the first stage puts out a first intermediate version of the program. Then you put this version into a second stage, which puts out a second intermediate version; and so on through various stages. This is done fairly automatically on big computers, but on little machines it's a pain.

(In fact, compilers tend to be very *slow programs*; but that depends on the amount of "optimizing" they do, that is, how efficient they try to make the object program.)

An *interpretive language* works differently. There sits in core a processor for the language called an *interpreter*; this goes through the program one step at a time, actually carrying out each operation in the list and going on to the next. TRAC and APL are interpretive; it's a good way to do quickie languages.

An Interpreter carries out,

A Compiler sets up.

Interpreters are perhaps the easier method of the two to grasp, since they seem to correspond a little better to the way many people think of computers. That doesn't mean they're better. For programs that have to be run over and over, compiling is usually more economical in the long run; but for programs that have to be repeatedly changed, interpreters are often simpler to work with.

A Black Art

Making language processors, especially compilers, is widely regarded as a black art. Some people have tricks that are virtual trademarks.

Actually, the design of a language, especially the *syntax*, how its commands fit together, strongly influences the design of its processor. BASIC and APL, for instance, work left to right on each line, and top to bottom on a program. Both act on something stored in a work area. TRAC, on the other hand, works left to right on a text string that changes size like a rubber band. Other languages exhibit comparable differences.

Mixed Cases and Variations (for the whimsical)

There are a lot of mixed cases. A *load and go compiler* (such as WATFOR) is put into

the computer with the program, compiles it, and then starts it going immediately. An *interpretive compiler* looks up what to do with a given instruction by interpreting it into a series of steps, but compiling them instead of carrying them out. (A firm called Digitek is well known for making very good compilers of this type.) An *incremental compiler* just runs along compiling a command at a time; this can be a lot faster but has drawbacks.

THE GREAT COMPUTER LANGUAGES

A certain number of computer languages are very widely accepted and used; I list them here. If you want to learn any of them, I believe that Daniel McCracken has written a manual on every one of them. (Not the variants listed, though.)

Why their names are always spelled with capital letters I don't know. (Generally they get let down in longer articles, though.)

Good Old FORTRAN

FORTRAN was created in the late fifties, largely by John Backus, as an algebraic programming system for the old IBM 704. (However, the usual story is that it stands for FORmula TRANslator.)

Fortran is "algebraic," that is, it uses an algebraic sort of notation and was mostly suited, in the beginning, to writing programs that carried out the sorts of formulas that you use in high school algebra. It's strong on numbers carried to a lot of decimal places (scientific numbers) and the handling of arrays, which is something else mathematicians and engineers do a lot.

Fortran has grown and grown, however; after Fortran I came Fortran II, Fortran III and Fortran IV; as well as a lot of variants like Fortran Pi ("irrational, and somewhere between III and IV"), WATFOR and WATFIV.

The larger Fortrans, that is, language processors that run on the bigger computers, now have many operations not contemplated in the original Fortran, including operations for handling text and so on.

BASIC is in some respects a simplified version of Fortran.

ALGOL LOST, AND PL/I

ALGOL is considered by many to be one of the best scientific languages; it has been widely accepted in Europe, and is the standard publication language in which procedures for doing things are published in this country. It is different from FORTRAN in many ways, but a key respect is this: while in FORTRAN the programmer must lay out at the beginning of his program exactly what spaces of core memory are to have what names, in ALGOL the spaces in core memory are not given names except within subsections of the program called procedures. When the program gets to a specific procedure, *then* the language processor names the spaces in core memory.

This has several advantages. One is that it can be used for so-called "recursive" programs, or programs that call new versions of themselves into operation. I guess we better not get into that. But mathematicians like it.

Originally this language was called IAL, for International Algebraic Language, but then as it grew and got polished by various international committees it was given its new name. (I don't know if anyone consciously named it after Algol, the star.)

It has gone through several versions. Algol 62, the publication language, is one thing; Algol 70, the 1970 version, is much more complicated and strange.

Several versions of ALGOL have gotten popular in this country. One, developed at the University of Michigan, is called MAD (Michigan Algorithm Decoder); its symbol is of course Alfred E Newman. Another favorite (for its name, anyway) is JOVIAL (Jules' Own Version of the International Algebraic Language), developed under Jules Schwartz (and supposedly named without his consultation) at System Development Corporation. JOVIAL is big in the US Air Force.

When IBM announced its System 360 back in 1964, there had been hope that they would support the international language committees and make ALGOL the basic language of their new computer line. No such luck. Instead they announced PL/I (Programming Language I), a computer language that was going to be all things to all men.

In programming style it resembled COBOL, but had facilities for varieties of "scientific" numbers and some good data structure systems. It is available for the 360 and for certain big Honeywell computers; indeed, the operating system for MULTICS at Massachusetts Institute of Technology was written in PL/I. Whether there are people who love the language I don't know; there are certainly people who hate it.

YECCCH, IT'S COBOL

Research and hobby types hate COBOL or ignore it, but it's the main business programming language. Your income tax, your checking account, your automobile license, all are presumably handled by programs in the COBOL language.

COBOL, or COmmon Business Oriented Language, was more or less demanded by the Department of Defense, and brought into being by a committee called CODASYL, which is apparently still going. COBOL uses mostly decimal numbers, is designed basically for batch processing and uses verbose and plonking command formats.

Just because it's standard for business programming doesn't mean it's the best or most efficient language for business programming; I've talked to people who advocate business programming in FORTRAN, BASIC, TRAC and even APL, But then you get into those endless arguments . . . and it turns out that a large proportion of business programmers only *know* Cobol, which pragmatically settles the argument.

There are people who say they've discovered hidden beauties in COBOL; for instance, that it's a splendid language for complex pointer manipulation. That's what makes horse racing.

JCL

Some call it Despicable, Some call it Home

"After you study it for six months, it makes perfect sense." — An IBM enthusiast.

JCL is a language with which you submit programs to an IBM 360 or 370 computer. "Submit" is right. Its complications, which many call unnecessary, symbolize the career of submission to IBM upon which the 360 programmer embarks.

SNOBOL

SNOBOL is the favorite computing language of a lot of my friends. It is a list-processing language, meaning it's good for amorphous data. (It derives from several previous list-processing languages, especially IPL-V and COMIT.)

SNOBOL is a big language, and only runs on big computers. The main concept of it is the "pattern match," whereby a string of symbols is examined to see if it has certain characteristics, including any particular contents, relations between contents, or other variations the programmer can specify; and the string substitution, where some specified string of symbols is replaced by another that the programmer contrives. [Who'll be the first person to simplify it and implement a microcomputer SNOBOL? . . . CH]

LISP

LISP is probably the favorite language of the artificial-intelligence freaks. A fondness for LISP, incidentally, is not considered to reflect on your masculinity.

LISP is a "cult" language, and its adherents are sometimes called Lispians. They see computer activities in somewhat different light, as composed of ever changing chains of things called "cars" and "cudders," which will not be explained here.

LISP was developed by John McCarthy at MIT, based largely on the Lambda notation of Alonzo Church. It allows the chaining of operations and data in deeply intermingled forms. While it runs on elegant principles, most people object to its innumerable parentheses (a feature shared to some extent by TRAC Language).

Joseph Weizenbaum, also of MIT, has created a language called SLIP, somewhat resembling LISP, which runs in FORTRAN. That means you can run LISP-like programs without having access to a LISP processor, which is helpful.

THEN, THERE'S ALWAYS MACHINE LANGUAGE

If you feel like making programs run *fast*, and not take up very much core memory, you go to machine language, the computer's very own wired-up deep down system of commands. It takes longer, usually, but many people consider it very satisfying.

Then, of course, if you have a particular style and approach and set of interests, you will probably start building up a collection of individual programs for your own purposes.

Then you'll work out simplified ways of calling these into operation and tying their results and data together.

Which means you'll have a language of your own.■

How to Build a Memory With One Layer Printed Circuits

Don Lancaster
Synergetics

The 2102 static programmable random access memory is a fairly obvious integrated circuit to use for a memory. It is easy to interface to just about any microprocessor or minicomputer. It costs from 0.1 to 0.3 cents a bit, buying from ads in BYTE or from lots of other possible sources.

What's not obvious is how you physically connect a group of eight or more 2102s for a 1 K x 8 or larger memory. We have ten address lines, a write line, an enable line, and two supply runs that have to go in parallel to each and every package. At the same time, separate input and output leads have to be provided in series for each different IC.

One elegant and very compact layout method is to use double sided, plated through boards and leads routed between IC pins. The trouble with this method is that plated through boards are extremely expensive, and there's no reasonable way to manufacture them on your kitchen table. Worse yet, the plating of holes makes removal of soldered parts very difficult, and the close tolerances of routing leads between pins invites trouble from solder splashes. Using double sided boards without plate through holes is even worse, since you have to solder top and bottom, and use of sockets or Molex *Soldercons* gets very ugly, if not downright impossible.

Single sided layouts, of course, are out of the question since they take far too many jumpers. Or do they?

Single Sided Layouts

Here's a simple technique that lets you build virtually any memory you want, using easy home brew low technology single sided boards without routing connections between IC pins, using piggyback ICs, or similar hassles. You can use direct soldering, Soldercons, or sockets per your choice. Whatever method you pick, the ICs are easy to install, test, and replace. The only penalty this method has is that you pay around 50% extra in the way of board area. And believe it or not, all it takes is *six* jumpers. And two of these are for convenience and can be eliminated.

The trick to all this is to pick very carefully what we call a jumper. Figure 1 shows the secret. Four of our "jumpers" are small strips of double sided PC board, 0.2 inch (0.508 cm) high and 5 inch (12.7 cm) long. The foil is somehow selectively removed so that each side touches the edge at only eight places. You can do this by etching, filing, carving, nibbling, scribing, chewing, routing, punching, notching, or just about any way that's convenient. Or, if you don't like using PC material for jumpers, you can use back to back insulated metal strips (Rodgers bus strip style), or you can forget

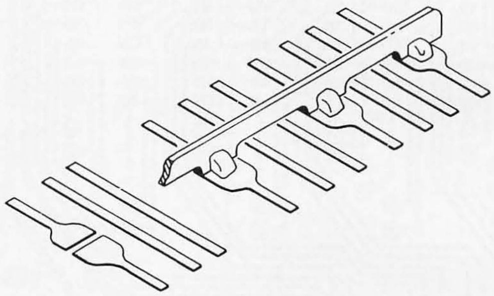


Figure 1: The secret of low cost single sided "no hassle" memory printed circuit layouts is jumper strips made from double sided PC material. Only four such jumper strips are required for the 1 K by 8 memory circuit. Pads on the printed circuit are arranged so that one electrical bus is connected to each side of the jumper strip. Electrical and mechanical connection of bus strips to the printed circuit is accomplished using fillets of solder.

the whole thing and use a Vector wiring pencil or an Applied Solder-Wrapper. What you end up with is one jumper strip that does the job of 14 individual jumpers (seven on each side).

Figure 2 shows us the schematic of a typical 1 K x 8 memory that uses 2102s. We've left it unbuffered for simplicity and low cost. Some background information on the 2102 is found in a separate box accompanying this article.

Our 1:1 PC layout is shown in figure 3, along with the pattern for one side of a typical bus strip jumper. The board has 25 mil (0.64 mm) lines on 25 mil (0.64 mm) centers and no routing between IC pins. With reasonable care, you should be able to handle this on a kitchen table PC lab setup. (I use a kitchen stove myself.) The output pins are conveniently grouped to a separate supply, address, data, and control runs as shown in figure 4. Two jumpers are used to

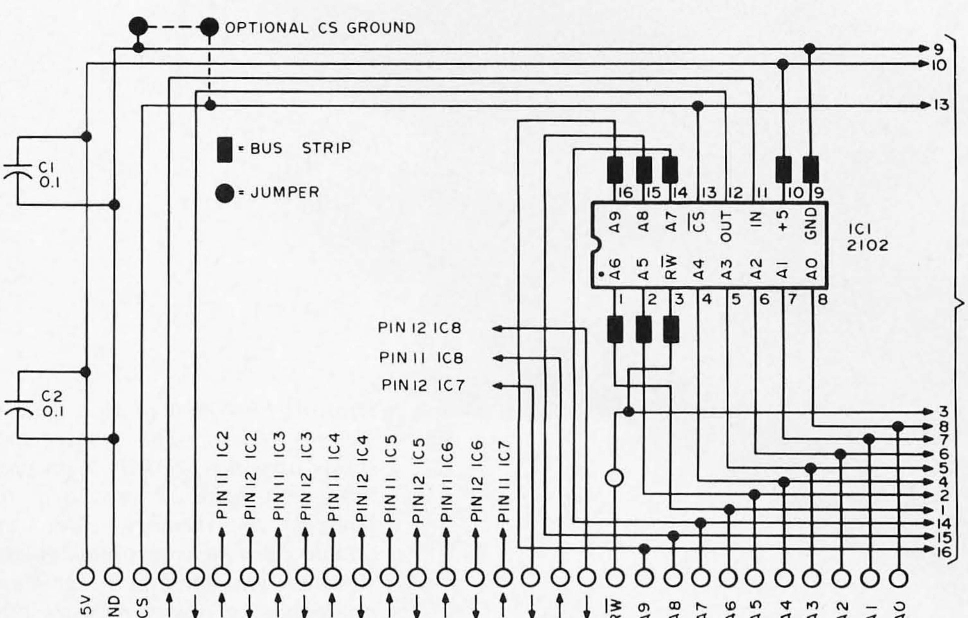


Figure 2: Schematic of a 1 K by 8 Memory Module. This diagram shows one of the eight 2102 memory circuits; IC2 to IC8 are connected in parallel.

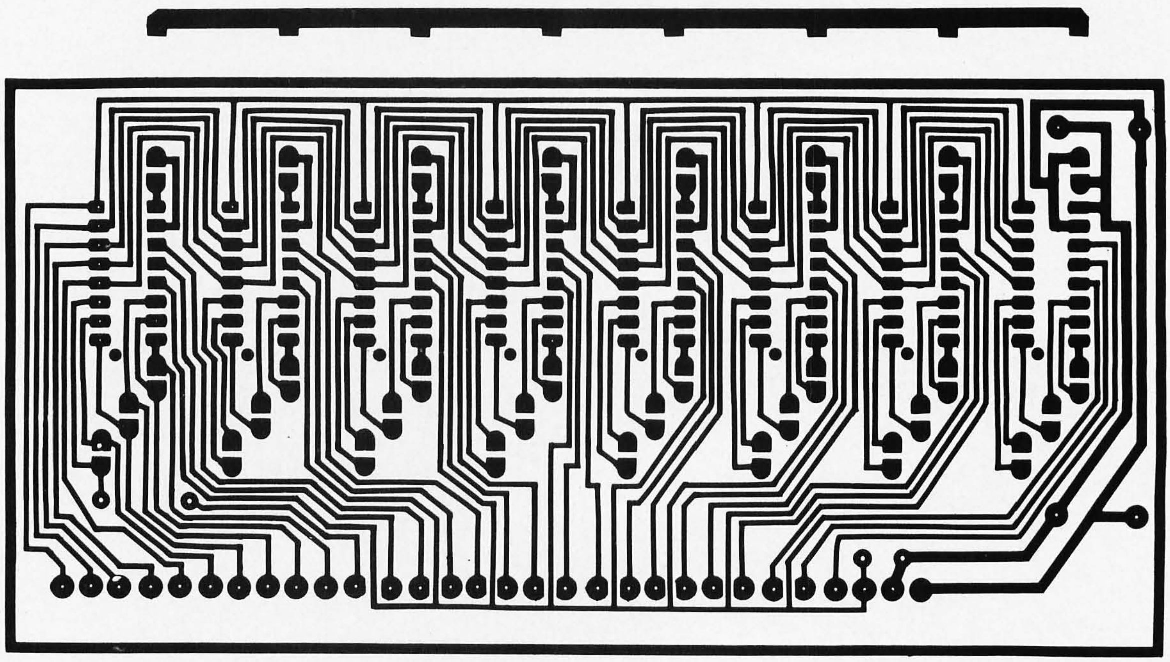


Figure 3: One to one PC Layout Pattern. The 1 K by 8 memory module can be fabricated at home using this pattern. The jumper strip silhouette is shown at the top of the figure. When drilling holes after etching, use the following sizes: #67 drill for the 128 IC pin holes; #60 drill for the four holes used to mount two bypass capacitors from +5 volts to ground; 0.0625 inch (1.5875 mm) drill for the 30 holes used to mount Molex connectors at the edge.

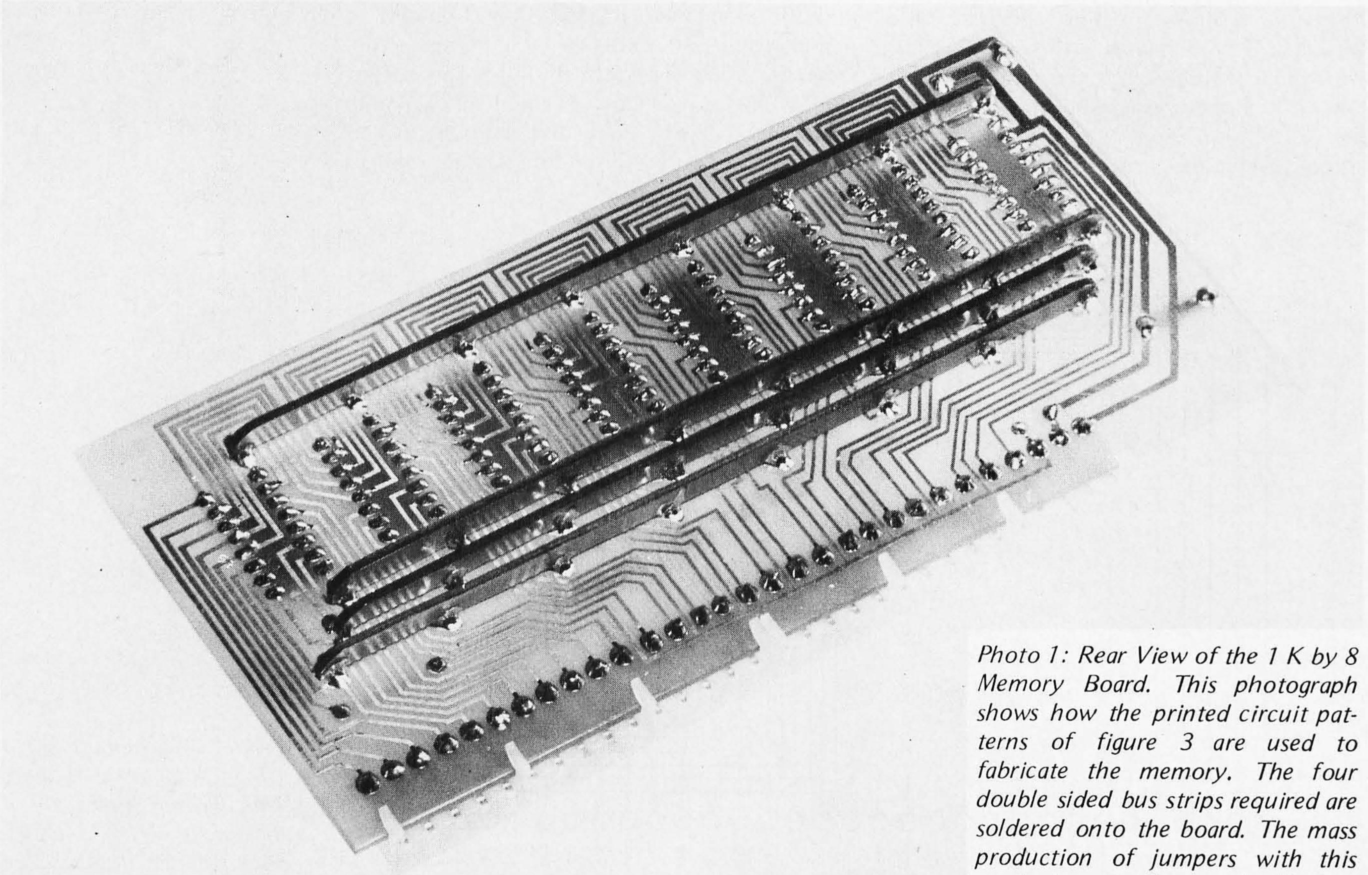


Photo 1: Rear View of the 1 K by 8 Memory Board. This photograph shows how the printed circuit patterns of figure 3 are used to fabricate the memory. The four double sided bus strips required are soldered onto the board. The mass production of jumpers with this method saves time and effort in construction of the board.

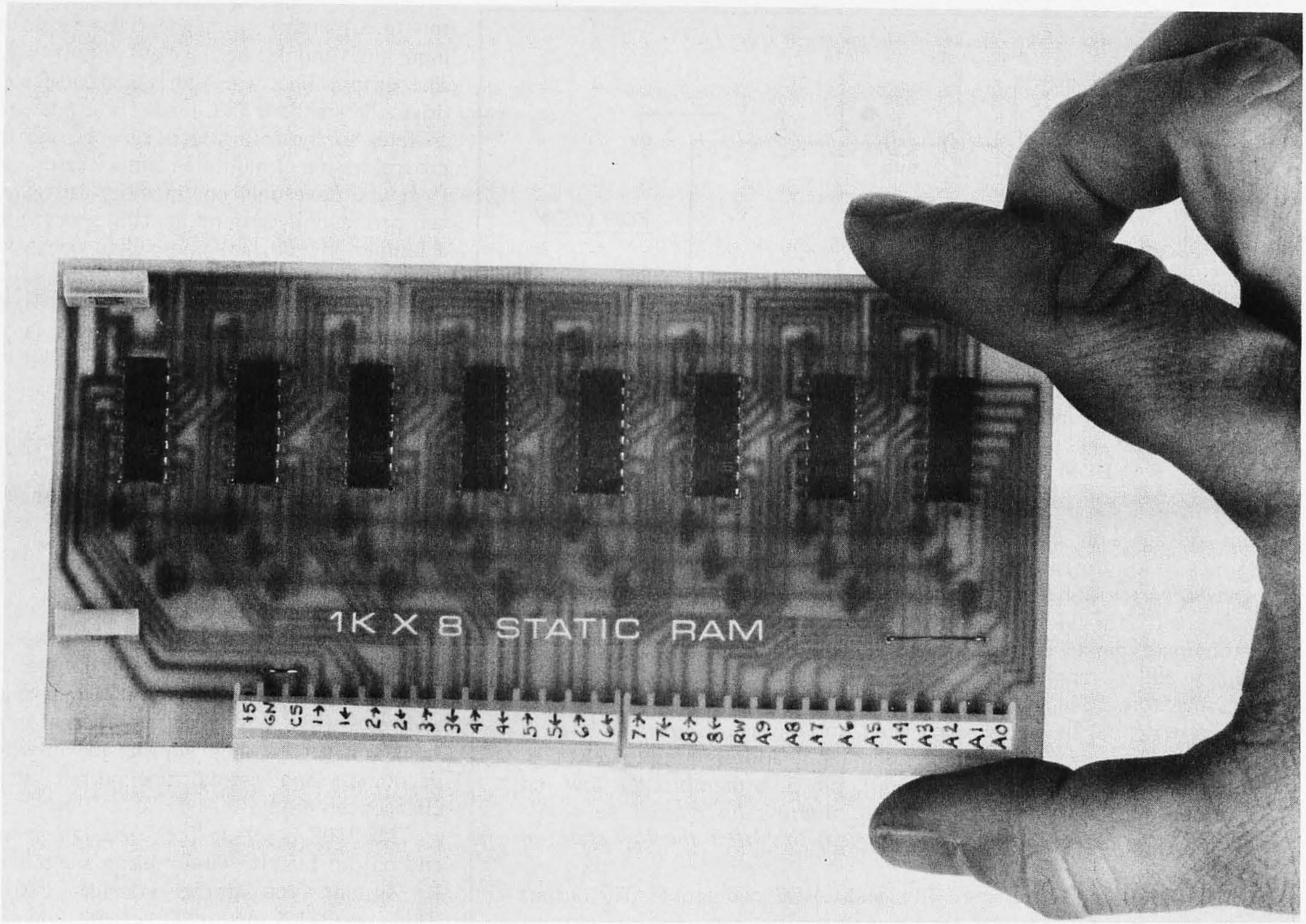


Photo 2: Front View of the 1 K by 8 Memory Board. The Molex edge connectors run along the bottom of the board. In the hand printed notations on the connectors, arrows indicate the direction of data flow for the data pins numbered 1 through 8.

line up the Chip Enable and Write pins in sensible places. Two 0.1 uF capacitors provide supply decoupling.

This particular layout is for 2102s, but you can easily use the same technique for 2101s, 2111s, 2112s, 5101s, dynamic memory, or just about anywhere else you have to connect lots of parallel leads to a bunch of ICs. The edge connector layout has been set up for Molex sockets or socket pins on 0.156 inch (0.396 cm) centers. You can use the sockets at the bottom as shown in photo, or the pins can be set up for a stacked board arrangement according to your needs.

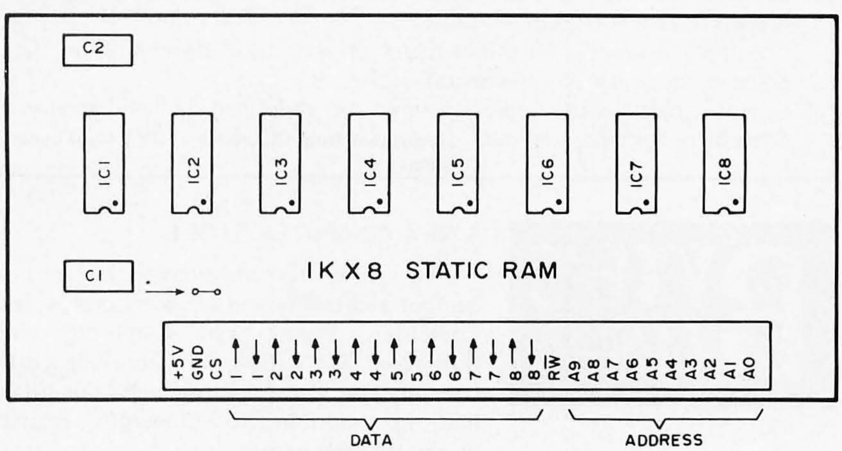
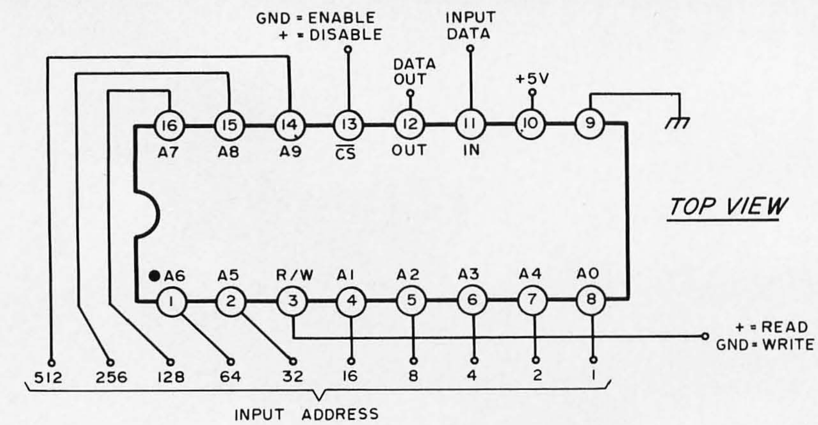


Figure 4: Component Placements. This diagram shows the location of the eight integrated circuits, two power supply bypass capacitors and edge connector pins. The arrow (*) identifies the optional chip select jumper wire.

Using It

The memory will need +5 volts at half an amp (much less if you use premium low power or CMOS RAMs). The ten address lines A0 through A9 select one of the available 1024 words. You can redefine these address lines any way you want. The only thing that's important is that each bit in the word sees the same address at the same time. Our special jumpers have done



STATIC RANDOM ACCESS MEMORY 2102

This is a static random access memory organized 1024 x 1 in a 16 pin package with separate input and output leads. Information may be rapidly read into and non-destructively read out of memory at system speeds. No clocking or refresh is needed. Storage is volatile with data being held only so long as supply power is applied.

To read, pin 13 is given a low level and pin 3 is given a high level. A binary address applied to the ten input address pins will select an internal storage cell and output the data in that cell.

To write, pin 13 is given a low level, and pin 3 is given a high level. A binary address is applied to the ten input address pins to select an internal storage cell. The write input, pin 3, is then brought low and returned high (a "write pulse"). *All address lines must be stable immediately before, during, and immediately after the low state on pin 3.*

All inputs and outputs are TTL and CMOS compatible. The output will drive one TTL load. Making pin 13 positive will float the outputs and ignore write commands. Outputs from separate devices may be connected in parallel so long as only one circuit is enabled at a time.

Access time varies with the manufacturer and the grade of the device. A 800 nanosecond read time and a 400 nanosecond write pulse is typical for a non-premium unit.

Supply power is 70 milliamperes or less, again depending on the grade of the device and the manufacturer.

Note that input addresses may be redefined in any manner convenient for circuit layout. **[Reprinted from Chapter 2 of TVT Cookbook (Sams).]**

this for us already. Data goes in on the eight input lines and out on the eight output lines. The output lines are TTL compatible and drive one standard TTL load. These lines are tristate, so the chip selects can be used to control busing of multiple memory cards.

If you have only one memory card, you can permanently ground the chip select with a jumper. If you have more than one, you have to be sure to ground and enable only one memory card at a time. The tristate memory outputs let you connect many cards in parallel, so long as you enable only one at a time.

The WRITE input should normally be held high. To enter data into memory, briefly bring this input low. The minimum write time depends on the 2102 you're using. Typical minimum times range from 300 to 700 nanoseconds. Check the data sheet for your particular IC.

One very important detail you will want to watch for: Be sure all your address inputs are stable immediately before, during, or after a write pulse. If you try to change addresses while writing, certain internal locations will get "flashed" during the internal decoding, and some unexpected data changes can result.

The 2102 is set up for a separate input and output bus. If you are using a common IO system, you should consider 2101s, 2111s, or 2112s instead. Otherwise, you can add an external bus transceiver such as an AMD 26S10, a TI 75138 or a Motorola 3443. A typical transceiver setup was shown in figure 7, page 17, BYTE No. 3. Note that several cards can share a single bus transceiver so long as you ground only a single chip select at a time. Additional address inputs and a suitable decoder can be used as a card or a page select. ■

BYTE'S BITS

A NEW COMPUTER STORE

One of the latest additions to the ranks of computer retailing and service stores is The Computer Store, 120 Cambridge St, Burlington MA 01803. The store opened in late January, and initially carries the Altair line in addition to numerous related products and services.

A Software Note from Processor Technology: FOCAL™ Language Release

Effective February 15 1976 Processor Technology Corporation released a version of the Digital Equipment Corporation's FOCAL language implemented to run on 8080 based microcomputers. FOCAL, a registered trademark of DEC, is an interactive interpretive language similar to BASIC. Included with the language are high level mathematical functions such as sine and tangent functions.

Cost of a paper tape of the language's object code is $3 and is part of a nationwide dealer promotion package. A source listing is also available as a part of the package, with the conditions of distribution set by the individual dealer.

Processor Technology has also released a resident Assembly Language Operating System, known as Software Package #1, which is available in source form for $3. Contact the factory or your local microcomputer dealer for further details.

Processor Technology is located at 2465 Fourth St, Berkeley CA 94710.

P173 Wiring Pencil

Both the hobbyist and the professional can save hours of wiring time with a new wiring pencil from Vector Electronic Company. Designated the P173, the pencil eliminates wire measuring, stripping, and forming. Average time to make a connection is only a few seconds.

The "pencil" consists of a feather-light plastic housing, a replaceable bobbin containing 250 feet of 36 gauge wire, and a tip to guide and cut the wire. The wire is routed directly between terminals or component leads making three or more wraps wherever a connection is to be made. Correct tension is maintained by the finger and the wire may be cut with the tip when the run is completed. Because the wire is insulated with polyurethane or nylon polyurethane, routing may be point-to-point without regard to possible shorting on intervening terminals. When heat is applied to the connection points with a fine-tipped soldering iron, the insulation melts, allowing a solder bond.

We'd like to see a detailed photographic article on this method of interconnection, comments on the electrical properties of small wires, hints and kinks for the experimenter. This method is another example of Vector's innovative interconnections work, and it is certainly one of the most attractive propositions for the hobbyist on a tight budget.

The tool is available from Vector Electronic Company, Inc, 12460 Gladstone Av, Sylmar CA 91342, or from several sources catering to the hobbyist.

MUSICOMP 76

The school of Advanced Technology of the State University of New York at Binghamton NY 13901, under a grant from the National Endowment for the Humanities, is sponsoring a workshop in computer techniques for music research, July 7–23, 1976.

Music historians, theorists, and analysts will be taught how to represent musical data in computer-readable form, fundamental concepts of computing, and PL/I. (The use of computers in musical composition will *not* be considered.)

An intensive workshop (July 14–17) in the DARMS (Digital Alternate Representation of Musical Scores) language and program support will be open to computer scientists, computer center personnel, and other interested persons even if they are not professional music scholars.

For detailed information write to MUSICOMP 76 at the address above.

Aargh!

(or, How to Automate PROM Burning Without EML)

Peter H Helmers
79 Evangeline St
Rochester NY

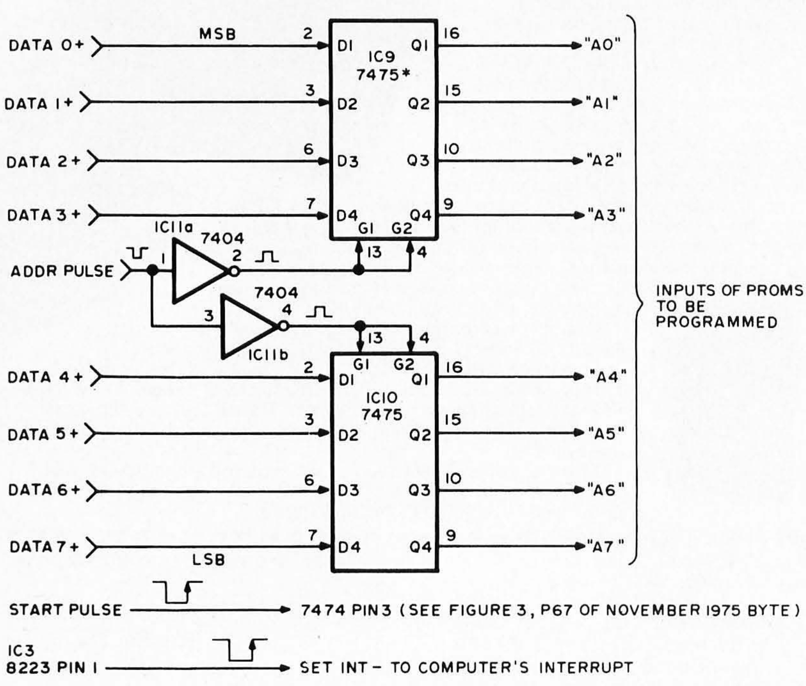


Figure 1: The word address for the PROM to be programmed may be easily set using a holding register loaded from the microcomputer through an output port. This register replaces the switches S4 to S11 in figure 1 on page 67 of the November 1975 BYTE. In this figure, ADDRPULSE- is a negative going signal generated by the output port of the computer when valid data for the PROM address is present at the lines DATA0+ through DATA7+. After the bit to be programmed is set up as described in figure 2 of this article, the programmer is started by means of a microcomputer generated STARTPULSE- signal which replaces the PROGRAM/VERIFY switch (S1a) in the original schematic of figure 3 on page 67 of the November 1975 BYTE. (S1b is replaced as shown in figure 3 of this article.) When the programmer's cycle is done, an interrupt should be set by means of the SETINT- line generated by the control ROM IC3 in the original design. If desired, the computer can verify the programmed bit by reading the output BIT- of the selected PROM line using the circuit shown in figure 3 of this article.

I just received my copy of BYTE No. 3 and quickly looked to find my article on the design of a PROM programmer. There it was, prominently featured on page 66. Oh well, I figured, hardware is always given lower priority that software. But then, I noticed an italic section of extensions and modifications had been added to my final thoughts. What could be put after *my final thoughts*???

I should not have read it. On the previous five pages was a design incorporating the glories of semiconductors, a design which was conceived to work well and reliably. And now this stranger, the editor of BYTE, had taken the liberty of suggesting modifying my design to use antiquated reed relays so that the programmer could be computer driven. Aargh! I believe that logic families work best if they are not mixed. Thus I submit the accompanying suggestions, in figures 1 through 3, which will interface a solid state microcomputer to the solid state programmer — using a solid state, not electromechanical, interface. The captions explain the modifications, so there is no need to elaborate any further on that subject.

However, I have been thinking: EML (ElectroMechanical Logic, sic) has some virtues. After all, it's well accepted and proven in the past several decades (newfangled bipolar logic is only 15 years old). And EML has tremendous noise immunity — both because of its high voltage swings (as much as 48 volts or more) and slow speed (it is totally immune to high frequency — greater than 1 kHz — noise). Finally, it offers large scale packaging.

Not being one to buck a winning trend, I offer the following ideas concerning EML. Once you have built the electromechanical interface to the PROM programmer, it will be easier to build your massive computer using EML, since you do not have to interface between logic families. I recommend a 16 accumulator machine architecture with

electromechanical indexed addressing. There are some good surplus buys on latching relays which you could use for memory. The program counter can be created from a stepping relay. However, limit possible jump instructions, since branching with a stepping relay may give rise to problems. And do not forget about mass storage. What is a disk read only memory, but a round piece of plywood mounted on a 33-1/3 RPM spindle and drilled out whenever you want a bit set to "1". A microswitch read head can be used to detect the presence or absence of holes. Finally, do not forget about the mechanical marvel that fits perfectly into this system — the good old Teletype! ■

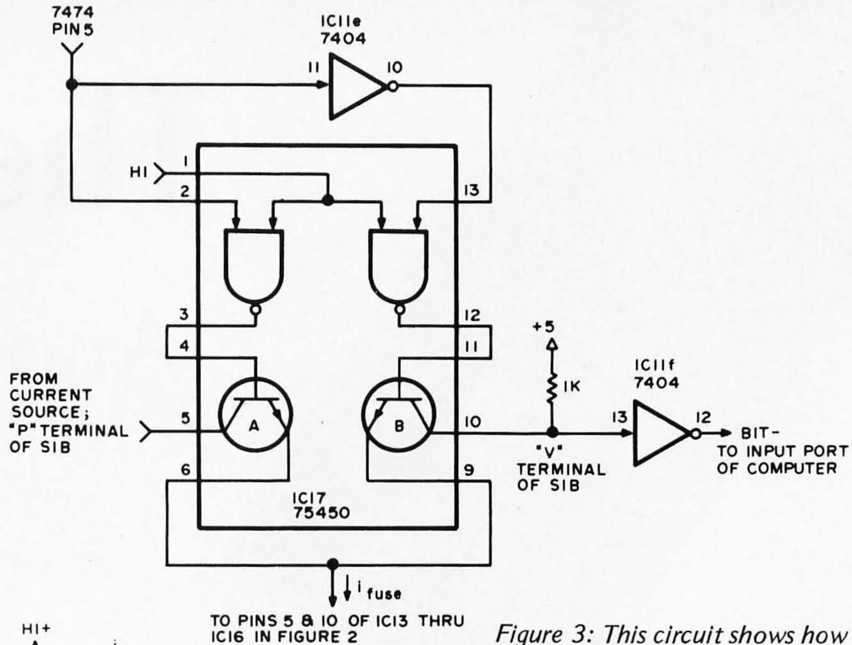


Figure 3: This circuit shows how the 75450 integrated circuit can be connected as a solid state replacement for S1b. What was the "V" terminal of S1b now goes to a 7404 buffer and is available as the signal BIT- which may be read by a one bit input port for verification purposes. The switching is controlled by the state counter enable flip flop (7474 IC4, pin 5). When this pin is low, transistor B conducts so that the selected output is available at BIT-. When IC4 pin 5 is high, transistor A conducts so that the fusing current i_{fuse} is applied to the selected output pin through one of the 75450 sections in figure 2 of this article.

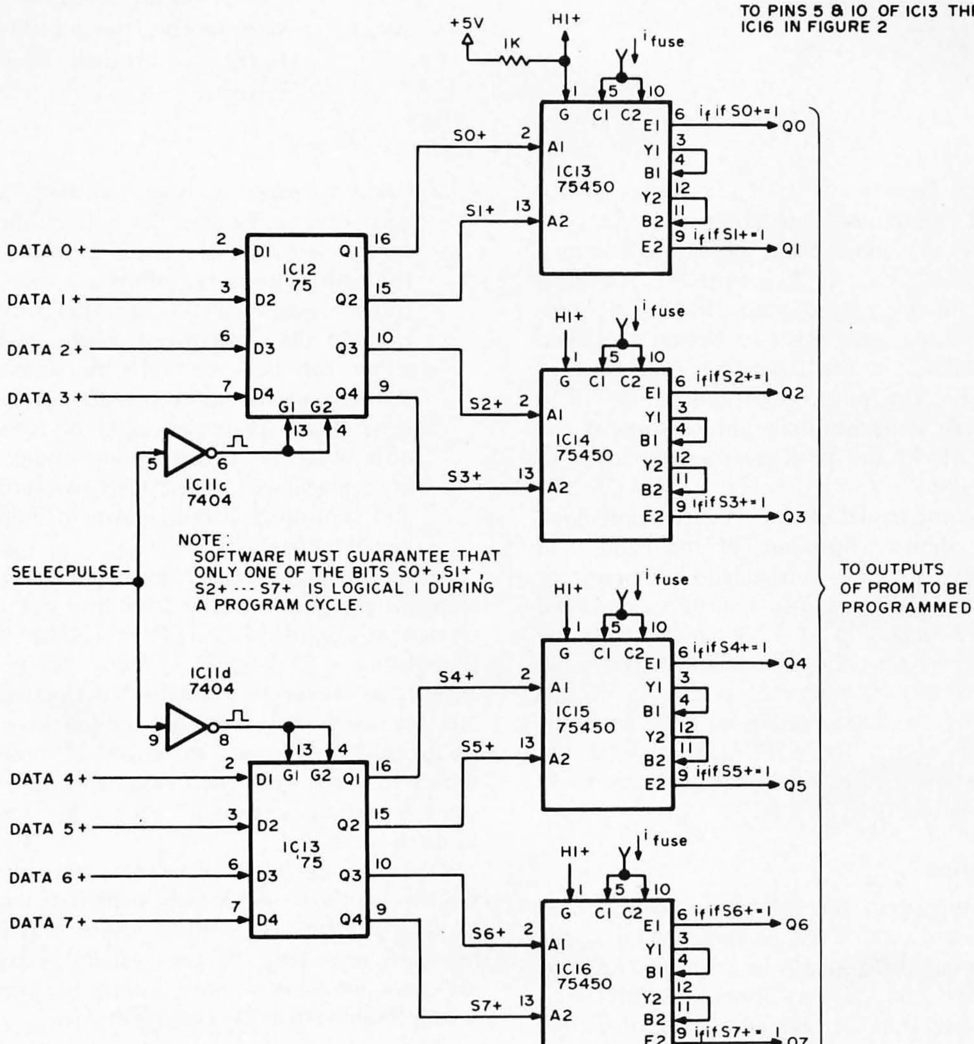


Figure 2: Data bus bits DATA0+ through DATA7+ are strobed into an output bit selection register by the microcomputer generated SELECPULSE - signal. Only one output should be selected (data bit set to logical one by the program) at a given time. The 75450 integrated circuit acts as a dual SPST switch to connect the i_{fuse} line to the selected PROM output and thus allowing verification and/or programming via the circuit of figure 3.

The SR-52: Another World's



The HP-65.

To the casual observer, the HP-65 and the SR-52 look very similar; the big difference is in their logic systems.

J Bradley Flippin
5044 Park Rim Dr
San Diego CA 92117

Using the parenthesis form of algebraic notation, the calculator's hardwired software analyzes the problem as stated in a "natural" form.

Infix notation has operators written *in between* two operands.

Postfix notation finds an operator following notation of two operands.

On September 16 1975 Texas Instruments announced the latest entry in their series of sophisticated pocket calculators, the SR-52, exactly 20 months after Hewlett-Packard announced their HP-65 fully programmable unit. Richard Nelson described the HP-65 in the December 1975 issue of BYTE. The purpose of this article is to provide some additional information on the new SR-52 and to provide a comparison to the HP-65.

To the casual observer the two units look very similar. Both are of the hand held variety, packing a tremendous amount of logic and memory into a small package. The SR-52 weighs in at 12.3 ounces while the HP-65 weighs only 11 ounces. One of the big differences is the retail price. As of this writing, the SR-52 retails for $395 while the HP-65 retails for $795 (although it can sometimes be obtained for $695 if one looks hard enough).

Notation

Other than the differences in price and keyboard (which will be discussed later), the other big difference is in their logic systems. Hewlett-Packard uses Reverse Polish Notation (RPN) in their line of pocket calculators while Texas Instruments has stayed with the algebraic (or infix) notation used by the rest of the calculator industry. It is interesting to read the literature because each company sets forth a very convincing case for its own system. Texas Instruments put it this way in their SR-52 flyer:

> "And to make it more confusing, a good case can be made for both by the careful selection of sample problems. In truth there is no ultimate answer. Either system can be operated with ease by the experienced owner. And either can be a boon to the simple solution of the most complex problems. Many practiced users of RPN now swear by it. But owners of algebraic machines can find RPN awkward and confusing. It boils down to individual preference."

Hewlett-Packard has an excellent paper comparing the two systems from their point of view; it's entitled "ENTER vs EQUALS" (Publication 5952-6035). There is no answer, as Texas Instruments has pointed out, because both systems get the job done. As a result, this is sure to be one of those topics that will keep their respective advocates trying to "convince" each other for years to come.

What are the main differences between the two logic systems? For the benefit of the reader, it might be well to spend a few moments describing the two systems (some say there are actually three systems because the algebraic system has two variations).

Algebraic (Infix) Notation

The simplest system for the novice is the algebraic system because it is the notation the average person has been taught in school. It is also known as infix notation. It is based on solving a problem in exactly the way it is

Smallest

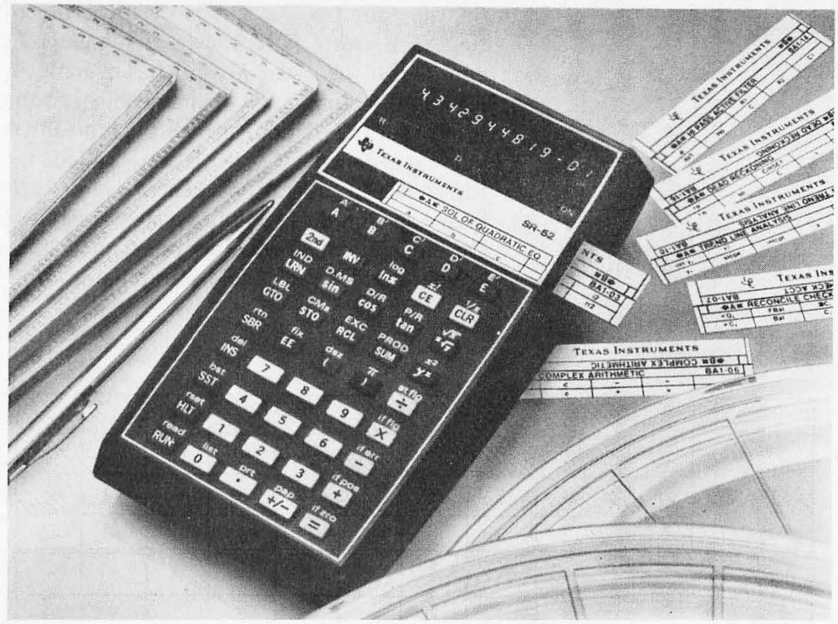


Photo 1: The SR-52 Calculator. Like its cousins in the microcomputer and large computer world, the SR-52 has magnetic recording features allowing the user to purchase and build a library of software.

commonly written. For example, let's solve the problem 5 + 4 = ?. The operator keys the data in exactly the way it is shown. He pushes "five", "plus", "four" and then hits the "equal" key and the answer appears in the display register. In reality the calculator's program maintains two internal registers, one of which is displayed. Pressing the first (five) key enters the data into the display register. Pressing the second (plus) key transfers the data to the second register which is also known as the accumulator. At this point the number "five" is in both registers and an internal switch has been set telling the logic that the next number is to be *added* to the accumulator when the time comes for the next arithmetic operation. The second number (four) is now entered into the display register. At this point the accumulator contains a "five"; the display register contains a "four" and the "plus" logic is set. To get the answer the operator simply presses the "equal" key which tells the logic circuits to perform the pending operation on the data in the accumulator, place the results in the display register and clear the preset function logic. Chain manipulations are possible by simply pressing another function key. Thus, the process can go on indefinitely. For example: 5 + 4 = 9 x 2 = 18 – 8 = 10 (etc.).

The operator can accomplish the same task more easily by eliminating the "equal" key each time. After "five", "plus", and "four", the operator could press the "plus" key directly. The same sequence described above will take place, except that the "plus" logic will again be set (actually, any function could be pressed). Thus, chain manipulations are possible in this way, also. For example: 5 + 4 x 2 – 8 = will provide the same results (10), but the number of steps needed is cut by two.

This is simple algebraic logic in that it does not use parenthesis to determine the precedence order from the entered data. A modification to the algebraic notation can be made by adding parentheses to help define the problem. For example: The problem 5 x 3 + 4 using simple chain manipulations will result in an answer of 19. However, if the operator wants to perform the summation first, then he would have to set it apart by parenthesis in this manner: 5 x (3 + 4). Now the answer is 35. Note also that the same result could have been obtained by rearranging the problem to read: 3 + 4 x 5. Now simple algebraic chain manipulation will result in an answer of 35 because the summation is performed first.

Parenthesis processing requires additional internal registers to hold the intermediate results. However, the process is similar to that described for the simple algebraic system. The only difference is that when the operator keys in the parenthesis, it tells the logic to "hold off" on the chain manipulation process until the matching parenthesis is found.

One might wonder why parentheses are needed because simple inspection of the problem will tell the operator that he must

A key to economical storage of programs in these small programmable machines is use of merged operations – two keystrokes which are stored as one location in memory.

These programmable calculators must surely be the ultimate in compactness and ingenuity in packaging.

do the "internal" portions of the problem first. The use of parentheses to some extent eliminates the need for such an analysis, leaving the breakdown of the problem to the internal logic. It does this through an internal precedence of calculations called the hierarchy of operations. There is no hierarchy in simple algebraic systems, because the logic simply processes the data as they are entered through chain manipulation.

Stack Content										
	T									
	Z						12	12		
	Y		3	3		12	5	5	12	
	X	3	3	4	12	5	5	6	30	42
Key		3	↑	4	x	5	↑	6	x	+
Step #		1	2	3	4	5	6	7	8	9

*Figure 1: An example of the HP-65 operations stack in use. This chart shows **numerical contents of the stack elements X, Y, Z, and T while calculating the problem (3 x 4) + (5 x 6). The HP-65 uses Reverse Polish Notation to calculate the result, using the keystrokes shown.***

Table 1: Detailed Comparisons. This table shows specific comparisons between the HP-65 and the SR-52 in areas of programming capability, calculating capability and operating characteristics.

Programming Capability	**SR-52**	**HP-65**
Program steps	224	100
Merged prefixes	all merged	stack and comparison
Merged store and recall instruction codes	no	yes
Program read/write	yes	yes
User-defined function keys	10	5
Possible labels	72	15
Absolute addressing	yes	no
Subroutine capability	yes	yes
Subroutine levels	2	1
Program flags	5	2
Unconditional branching	yes	yes
Conditional branching decisions	10	7
Indirect branching	yes	no
Editing		
Single-step	yes	yes
Back-step	yes	no
Insert	yes	yes
Delete	yes	yes
Single-step program execution	yes	yes
Optional lock-in printer	yes	no

A question one always asks when parentheses are encountered is how deeply may they be nested? (Remember, each parenthesized level in the problem requires additional internal storage for intermediate results). Some systems go four or five levels deep. The SR-52 is capable of nesting to nine levels. The following example illustrates the maximum capability of the SR-52 in this area:

6 x (9 + (6 x (12 ÷ (3 x (8 x (2 x (6 ÷ (6 x (6 + 2))))))))).

Straight chain manipulation, disregarding parentheses, will yield a result of 3458, which is incorrect. The correct answer is 4.5. It should be noted that the parenthesis count does not include expressions that have already been terminated. For example: 6 x ((5 + 7) ÷ (6 x 9)) is not three deep, but only two deep because the first expression (5 + 7) was terminated by the first right parenthesis, ")", prior to encountering the second expression (6 x 9). Internally, each level of parentheses is like using one level of the stack in a Reverse Polish Notation machine.

Reverse Polish Notation

The second logic system is known as Reverse Polish Notation (RPN). The Polish mathematician Jan Lukasiewicz wrote a book published in 1951 on formal logic wherein he was the first to demonstrate that an arbitrary expression could be shown unambiguously without the use of parentheses by placing the operators immediately in front of or after their operands. For example, (a + b) x (c – d) could also be expressed as x + ab – cd (keep in mind that ab is a logical notation and does not mean multiplication but shows only the sequence of the data). This is a prefix type of notation. It could also be reversed to provide a postfix type of notation as follows: ab + cd – x. (Now you know why the algebraic system in common use today is called infix notation, since the operators are in the middle between operands). As a result of this discovery, both prefix and postfix notation have become widely known, respectively, as Polish and Reverse Polish Notation in Lukasiewicz's honor.

Reverse Polish Notation, as mentioned above, does not utilize parentheses. As a result, the solution to problems using this type of logic must be approached in exactly the same manner as with any computer program because it is the same logic used in all large (and nowadays, small) computer systems. The programmer moves his data around into various registers and then, once

it is where he wants it, he executes the required arithmetic operation.

Hewlett-Packard has arranged its working registers into what they call an operational stack. It consists of four registers designated X, Y, Z and T. A fifth register is called the LAST X register and is a recent addition to the HP line, although it is not directly a part of the stack itself. The LAST X register holds the last data entry in the event the operator wants to either see what it was, wants to use it again, or wants to extract it from the solution (i.e., entered the correct data but pushed the wrong function. The operator simply presses LAST X, the inverse of the previous function and then the correct function).

The four operational stack registers have special functions and operate as an integrated group. The X register is the data entry register and is the only one that is displayed (For those who have used the HP-9100/9810 series, they displayed the X, Y and Z registers simultaneously). All of the trigonometric and some of the transcendental functions are performed directly in the X register (i.e., 1n x). The Y register can be thought of as the accumulator. All mathematical operations are performed in this register. Those transcendental functions which require two registers use both the X and Y registers (i.e., y^x). The Z and T registers are temporary storage registers and no mathematical operations can occur in them. Data are moved to and from them as required during the solution.

If one uses a concept such as an operational stack, it sometimes proves necessary to move the data around within the stack. To perform this function, Hewlett-Packard has designated a special set of data movement keys which do not appear (nor are they required) on the SR-52. For example, ROLL UP and ROLL DN allow the contents of the stack to be shifted up or down one position (in much the same manner as a circular shift or rotate instruction). When shifted up, the data in the T register goes into the X register and vice versa. The ENTER (or UP) key literally pushes the data up. Thus, the contents of all registers are moved up one, with two exceptions: The original data stays in the X register and the data in the T register is destroyed by the data from the Z register. As a result, if the operator desires, the same number can be placed in all registers by pressing the sequence: (data)(UP)(UP)(UP).* Figure 1 is an example of the use of an operational stack for the problem (3 x 4) + (5 x 6). The ENTER (↑) key breaks up the solution by moving the intermediate data up into the stack where it is saved until needed. In addition, the HP-65 has a special feature of automatically inserting an UP function prior to any data entry that follows a functional operation. This can be seen between steps four and five in figure 1. Notice that the intermediate result (12) moved up automatically as the five was entered. The operator must, however, ensure that he does not move up more than three intermediate results, which is the HP-65's limit (without using the data storage registers). In comparison, the SR-52 can handle up to ten pending operations.

As a further assist in manipulating data in

Table 1 (continued):

Calculating Capability

	SR-52	HP-65
log, lnx	yes	yes
10^x, e^x	yes	yes
x^2	yes	yes
$\sqrt{x}$	yes	yes
$\sqrt[x]{y}$	yes	no
y^x	yes	yes
1/x	yes	yes
x! (factorial)	yes	yes
Trigonometric functions	yes	yes
Degrees-minutes-seconds to decimal degrees conversion	yes	yes
Degree, minute, second arithmetic (+, –)	no	yes
Degree/radian conversion key	yes	no
Polar/rectangular conversion	yes	yes
Octal conversion	no	yes
Absolute value	no	yes
Integer, fraction part	no	yes
Built-in π value precision	12 digits	10 digits

Operating Characteristics

	SR-52	HP-65
Angular modes	2	3
Fixed-decimal option	yes	yes
Calculating digits	12	10
Digits displayed (mantissa + exponent)	10+2	10+2
Data memories	20	9
Memory arithmetic (+, –, x, ÷)	yes	yes
Exchange x with y	no	yes
Exchange x with data memory	yes	no
Entry mode	algebraic	RPN
Max. number of pending operations handled	10	3
Number of keys	45	35
Indirect memory addressing	yes	no

***NOTE: For examples of keystroke sequences, the key name or a description of input (such as "UP") is enclosed in parentheses.**

the stack, the HP-65 has an EXCHANGE X AND Y key ($x \rightleftarrows y$) which allows the operator to interchange the contents of these two important registers. This is handy when encountering operations where ordering of operands is vital, as in division. Using this operation, it is also possible to reverse the sequence of the data in the entire stack: ($x \rightleftarrows y$), (ROLL UP), (ROLL UP), and ($x \rightleftarrows y$); (ROLL DN) could have been used, if desired.

Function Selection

The next area of interest is the keyboard itself. The SR-52 has 45 keys where the HP-65 has only 35 keys. Both machines use the "second function" type of system which allows one key to have two meanings. The SR-52 uses the symbol "2nd" while HP uses "f". In addition, the HP-65 also has a "third function" key designated "g" which allows all of their keys to take on a third meaning. Both units have an inverse function key designated INV on the SR-52 and f^{-1} on the HP-65. This is handy for finding the ARC SIN of a number. The operator simply presses: (2nd) (SIN). A comparison of the photographs with this article will reveal other differences in the keyboard layouts.

Operation Codes

The key to economical use of storage in this type of machine is the application of merged operations which allow one storage location to accept a prefix, when required, along with an associated instruction code. This is possible in both units; however, only the HP-65 allows merged storage and recall instructions. A practical relationship between the two programming systems can be obtained by using the problem on page 74 in the *HP-65 Owner's Handbook* as a comparison. It is a financial interest problem containing 22 storage and recall instructions. The problem requires 68 memory locations in the HP-65, while in the SR-52 (neglecting any translation due to logic systems, which is left to the reader as an exercise), it would require 90 memory locations. The difference is due to the SR-52's lack of merged storage or recall instructions. This may be one of the reasons Texas Instruments made their program memory over twice as big as the Hewlett-Packard's.

Both units use a similar method of designating their decimal numeric instruction (or operation) codes. With the exception of the digit keys, the instruction codes for any key on either unit can be found by simply counting down the left column to that row and then counting across to the particular key. Thus, on the SR-52, the



And Now, a Printer for the SR-52

On January 7 1976 Texas Instruments Inc, Calculator Products Division, announced the new PC-100 print cradle for the SR-52 calculator. The product is a desk top unit with a 20 character per line thermal printer using 2.5 inch (6.35 cm) thermal printing paper available in roll form. The PC-100 interfaces to the calculator and expands capabilities to include program listing and execution trace capabilities. The listing allows a permanent human readable record of the program to be made automatically; a trace documents each calculation step of a program as it is performed. An extra feature is that the calculator can be locked into the base provided by the printer, making the entire system less likely to be pocketed by unscrupulous individuals. All this function is available for only $295, and the unit will be sold through the usual TI calculator distribution channels (direct mail and retail stores).

For further information on the PC-100, contact Texas Instruments Inc PO Box 5012, Mail Station 84, Dallas TX 75222 (Attn: PC-100).

instruction code for Enter Exponent (EE) is 52 while the same function on the HP-65 (EEX) is 43 (this can be verified from the two photographs). The digit keys retain their own values (i.e., one is 01, two is 02, etc.).

Comparisons?

There are so many features of both units that it is difficult to say any one of them is the *big* feature; however, the fact that they can both record and read programs on small magnetic cards certainly ranks high on the list. A detailed features comparison is found in table 1. Because of the SR-52's large memory, it requires two passes to read or write the card. The card is inserted in the A direction first and then turned around (not over, as the oxide must remain face down) and the B side is entered. Both units contain a recessed card holder between the display and the five special function keys which are labeled A through E. The cards have an area upon which the operator can write to designate the functions of the special keys for customized programs. In the case of pre-recorded programs, the data elements are also pre-printed on the cards as can be seen in the photograph.

Last, but not least, is the one big feature of the SR-52 which is not yet available with the HP-65 system. In early 1976 Texas Instruments intends to market the PC-100 which is an optional desk top lock in printer for use with the SR-52. It looks like a regular desk calculator with the typical adding machine type tape printing unit on the left and a space on the right for the SR-52. The unit includes a key lock so the calculator cannot be "lost." It will allow the user to list out entire programs, print the results of calculations, and advance the paper. These functions are already on the SR-52's keyboard as second functions (LIST, PRT, and PAP, respectively).

This short article has not covered all points of comparison between the two calculators. As can be seen from table 1, there are many areas that have not been discussed. The purpose has been to inform you about these two interesting computer systems. However, if you feel teased and want to investigate these fascinating machines further, the manufacturers would love to tell you where you can see them in your community. Both have toll free (WATS) numbers you can use: Texas Instruments (800) 527-4980 [in Texas (800) 492-4298], Hewlett-Packard (800) 538-7922, ext 1000 [in California (800) 662-9862]. These programmable calculators must surely be the ultimate in compactness and ingenuity in packaging.■

Controlling External Devices With Hobbyist Computers

Robert J Bosen
Box 93
Magna UT 84044

There is an almost infinite variety of uses to which a hobbyist computer system may be applied besides calculating or data processing, and many of these can bring a great deal of satisfaction to the proud owner. For example, hobbyist microcomputers are invariably advertised with a long list of possible applications such as home security systems, light controllers, process controllers, or automated drink mixers. I have personally had several opportunities to use my computer in a variety of related ways, including controlling stage lighting and sound effects for a large bicentennial celebration, and automating a spook alley. These and other applications inspired me to build the module described here to interface my computer with virtually any electrical or electronic device. If you build this interface as I did, you'll be able to control up to 16 channels of electrical outlets or switches of any kind, and only your imagination will limit the applications.

The basic principle behind any computer interface is to change computer compatible signals to device compatible power levels, and this interface accomplishes that goal with a great deal of flexibility, allowing the user to hook up virtually any type of

Photo 1: The author's computer setup includes the two CRT terminals shown on the table, plus a rack cabinet presently containing his central processor.



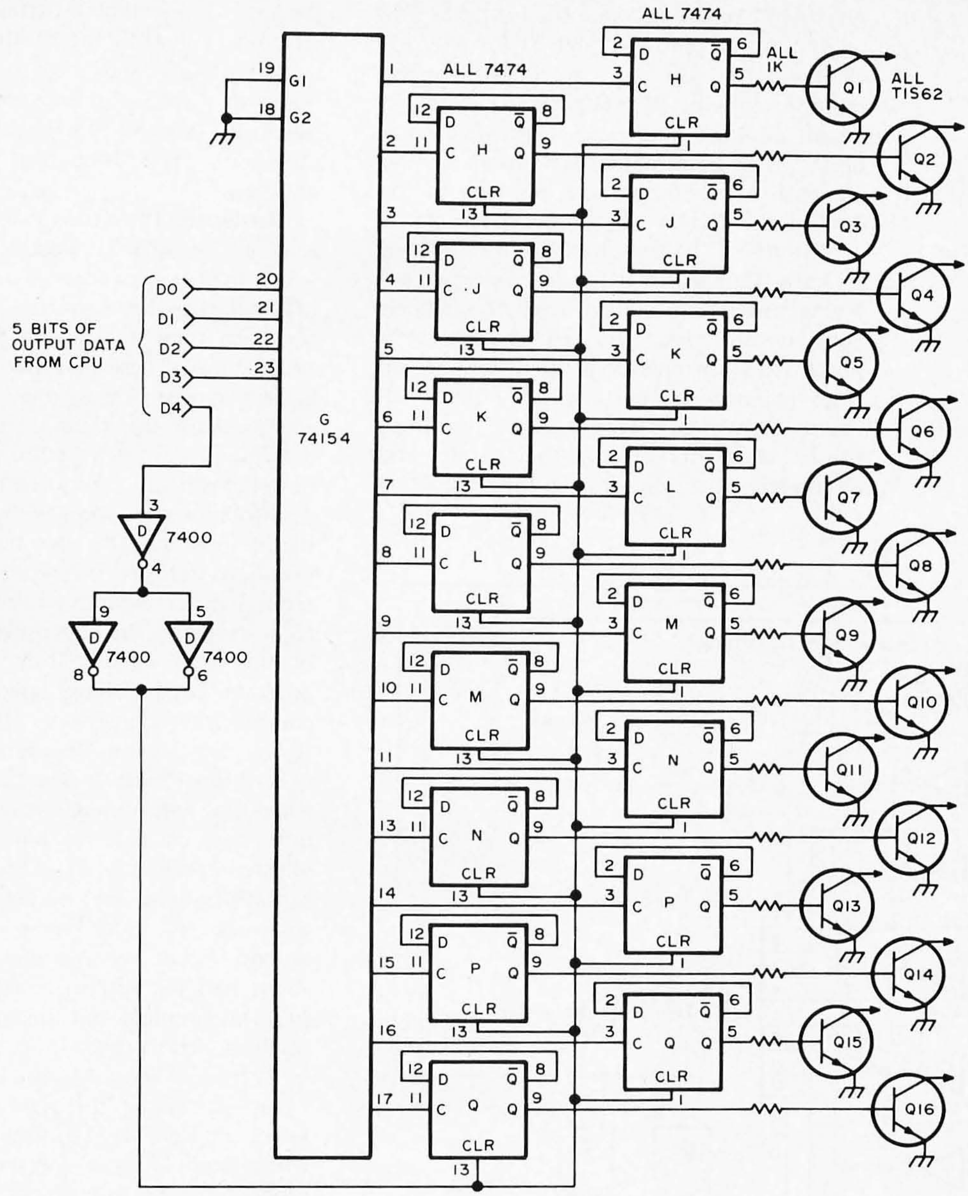


Figure 1: This is all you'll need to build if you already have a parallel output port you can use to control the interface card. If not, lines D0 through D4 should be joined with the corresponding points in figure 2. Transistors Q1 through Q16 can be any economical NPN with reasonable Beta. Due to varying configurations (you may not want to build up all 16 channels or use different transistors), I suggest the card be wirewrapped.

transistor, relay, or small electrical device to its open collector outputs. I used 16 surplus relays and wired them to 16 AC outlets and 16 sets of "five-way" binding posts. But this is by no means the only way to utilize the 16 output channels provided. All in all, the system described allows the programmer a great deal of flexibility over what he will control and how he will do it.

This interface may be used with virtually any 8 bit computer, and could be modified to work with a 4 bit machine as well. The circuit consists of four parts: A parallel output port, a 16 channel demultiplexer, a 16 bit memory, and 16 single transistor driver amplifiers. It can be built on a single small circuit board and total cost for all the solid state parts will be under $35 if a little shopping around is done. If you already have a spare parallel output port you can dedicate to this purpose, you can save about half of that cost.

Here's how it works: A byte of data is sent out of the computer to the parallel output port where it is latched. The four low order bits are applied to the four inputs of

the 74154 demultiplexer which selects one of 16 output pins and pulls it low. If, for example, the four bits are 0000, the demultiplexer will select channel zero and pin 1 will go low. There are 16 possible combinations of data that may be received, and for each of these combinations one of the pins of the 74154 will go low. Each of the 16 outputs of the demultiplexer then goes to a D flip flop which it toggles. Since we are trying to exercise control over 16 channels continuously, but the 74154 can only process one channel at a time, these D flip flops are needed to store the status of all inactive channels. Toggling the flip flops causes them to reverse their state and alternately turn on or off the transistors they drive each time a particular channel is selected. The fifth bit of the data byte is buffered (IC D) and then runs to the reset inputs of all 16 D flip flops, providing a reset signal to turn all the channels off simultaneously. (The three high order bits are unused.)

Hardware. The circuit provides 16 transistors in an open collector configuration, which may be viewed as open switches when off, and as switches shorted to ground when on. Each transistor can handle about 30 V and 30 mA. These may be used to control bigger transistors, or relay coils may be energized through them, or small electronic devices (sirens, light bulbs, etc.) may be powered directly with them by placing a voltage source in series with the device and the transistor. This is shown in several variations in figure 3. A word of caution is in order here if inductive loads such as relay coils are used: The collapsing magnetic field of the relay coil as it is turned off can generate large voltage spikes which may damage the transistors. Relay coils (see figure 3a) should therefore be protected with shunt diodes to short out these spikes when they approach dangerous levels. Relays may also oscillate at high frequencies if selected frequently in a program, so small capacitors may be necessary across the windings to short these oscillations to ground. From my own experience I found about half the surplus relays I tried exhibited this problem, but tinkering with various small capacitors clears it up.

Software. The software must provide data bytes containing the right information to select the right device at the right time. This will require a little forethought from the programmer because of the nature of the D flip flops used to store the status of each channel. Returning to the preceding discussion on circuit operation, it will be recalled that the D flip flops toggle (reverse states) each time they are selected. However, simply selecting the same channel over and over again will not toggle it on and off as it might

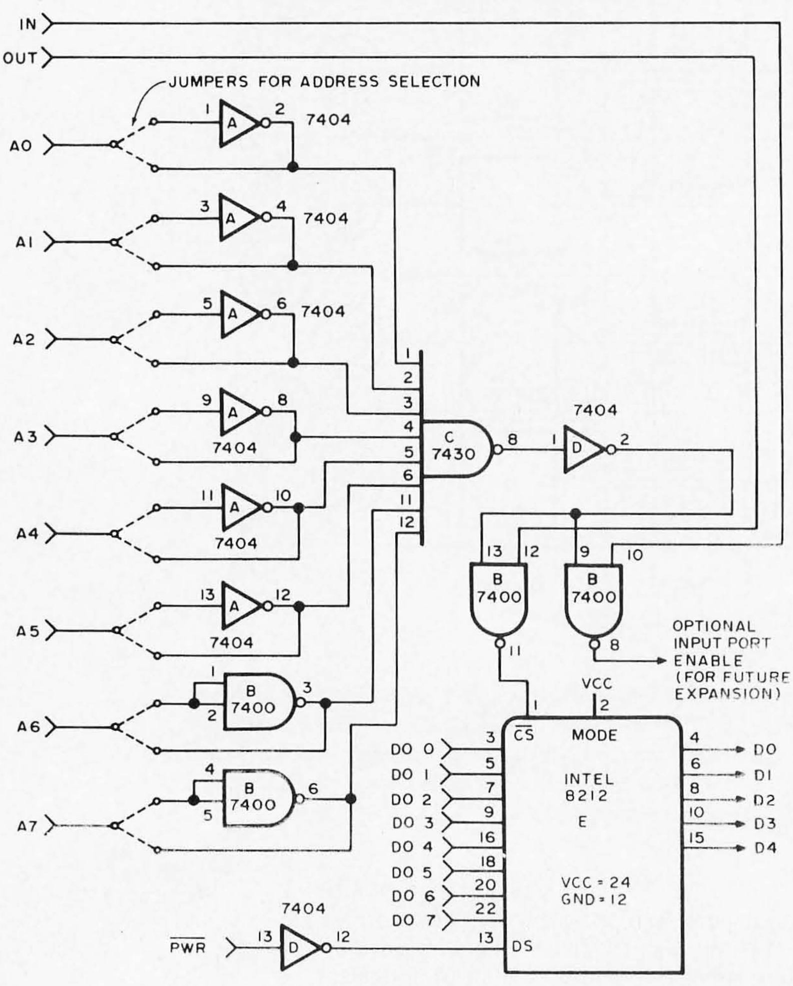


Figure 2: This is a standard parallel output port, capable of responding to any output address between zero and 255. The address is specified by the eight jumpers coming off the address lines. You may want to use low power chips (74L series) for IC A, IC B and IC C, to save on address bus loading. Incidentally, this addressed output port could be used in any application requiring a parallel output. All eight data lines are available at the various outputs of the 8212 chip. The IN and OUT and PWR inputs are for Altair 8800 and similar computers. The OPTIONAL INPUT PORT ENABLE line coming from pin 8 of IC B may be used to enable another 8212 chip with the CS pin to function as an input port and place data on the input bus when the IN line is active and the specified address is enabled.

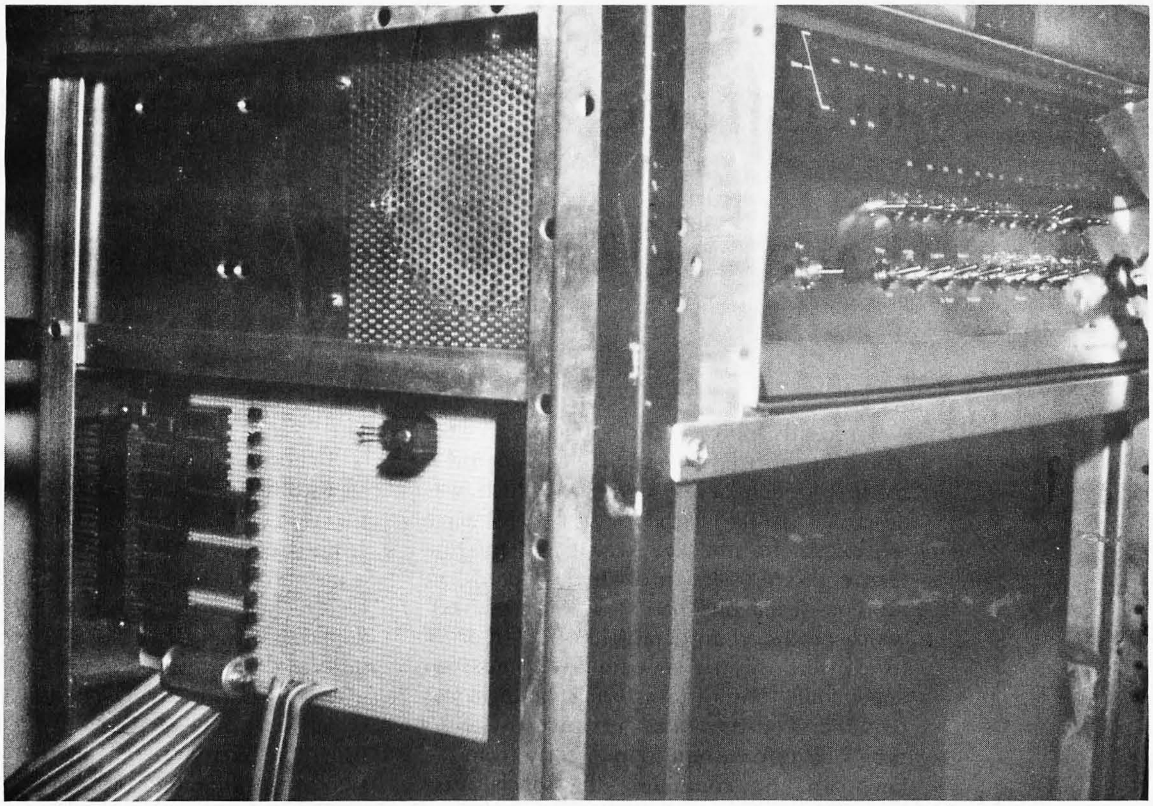


Photo 2: Details of the output control interface. The interface was built upon perforated board mounted at the side of the rack cabinet at the left.

be expected, because the D flip flops only toggle on *rising* edges from the demultiplexer, and a rising edge only occurs *after* a channel has been selected when the multiplexer *changes* to select (ground out) a different channel. So, turning a channel on and then off is accomplished by first selecting the desired channel with a data byte, then selecting a different channel (This might be an unused channel or the next sequential channel in your program), then waiting the delay needed for the first channel to switch on, then selecting it again to reset it. This may seem a little complicated at first, but it's easy to get used to.

Applications. Software and hardware will of course be determined by the application needed, and this will vary widely from instance to instance. The following ideas have occurred to me and you will undoubtedly think of many more: Light shows, computer music, industrial process control, computerized games, industrial robots, stage lighting, spook alleys (Electro-Spook?), slide presentations, darkroom automation, chemical mixing, remote controls of any type, or a fully programmable electrically operated teeter-totter. Try it – you'll like it!■

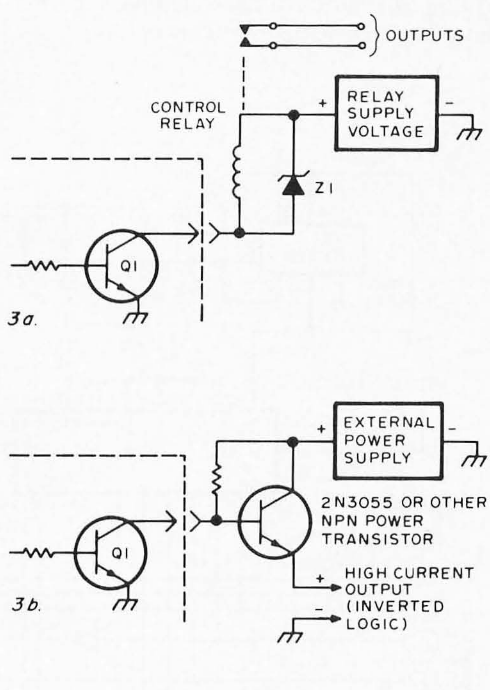


Figure 3a: Interfacing relays. Z_1 is used to protect Q_1 from spikes. Z_1 should have a breakdown voltage just higher than the relay voltage.

Figure 3b: Power transistor interface, suitable for powering tape recorders or other small appliances.

Figure 3c: Small load (<=30 mA) direct interface.

Interface an ASCII Keyboard to a 60 mA TTY Loop

Jay A Cotton
Bldg 844, Apt 2H
Gov Island NY 10004

I recently purchased a Sanders 720 electronic keyboard. This keyboard is identical to the Model 722-1 keyboard which was described in BYTE, September 1975, page 62, except for the key layout and the line feed code. My version of the keyboard had no line feed, but had a vertical tab key which produced an octal 013 code. In order to convert this to an octal 012 line feed code, some form of transformation logic was required. I also wanted to drive my Teletype's 60 mA current loop directly from the keyboard. By combining the special case code conversion, a UART for parallel to serial conversion, a clock and a current loop driver, I achieved the desired function of sending characters to my Teletype. Figure 1 shows the schematic of this conversion.

The Circuit

I chose to detect the octal code 013, then to use this special case to alter the data on the low order bit of the parallel code presented to the UART. By changing the low order bit of the octal 013 code from a logical one to a logical zero, the number is converted from 013 to 012. The 013 code is detected using inverters and the 7430 NAND gate shown in figure 1. The low order bit is selectively changed for this one code by using the exclusive OR function of one section of the 7486 integrated circuit. When the input at pin 2 is low (the normal case without the 013 code input), the exclusive OR normally passes line 0's value directly to the UART pin 26 input; when the input at pin 2 of the exclusive OR is high (as is the case

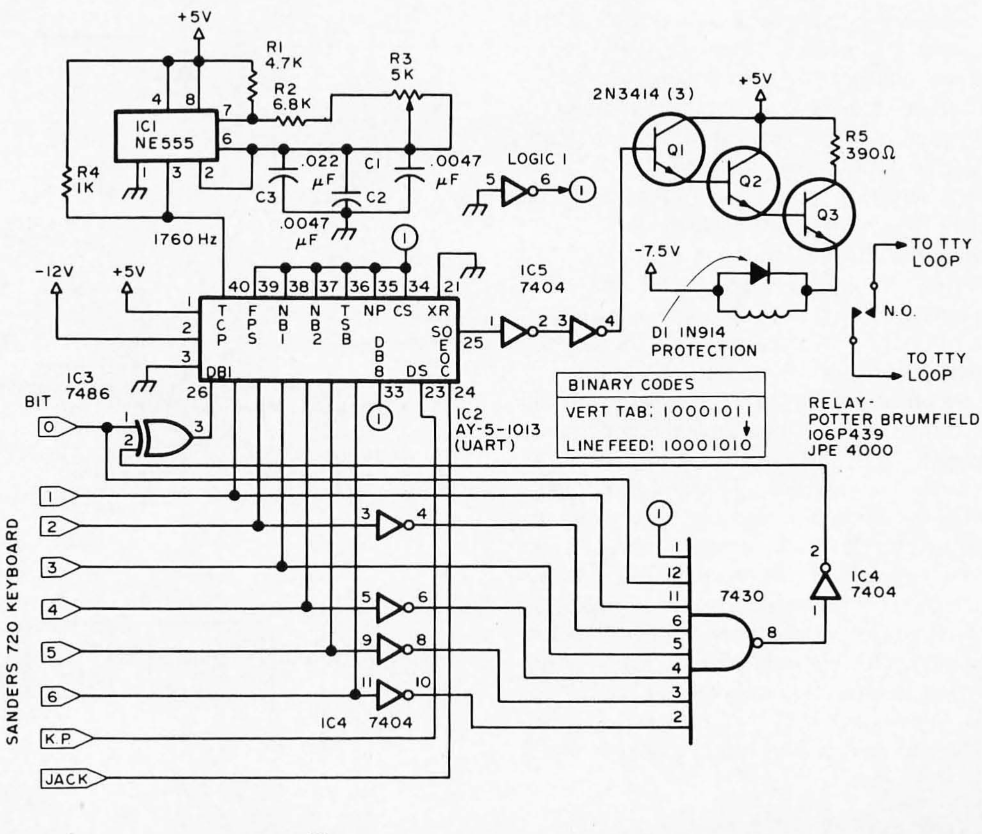


Figure 1: Using a UART and special case logic to convert and serialize the output of a keyboard for a 60 mA current loop.

when 013 is detected), the exclusive OR function inverts the value of line 0, thus transforming 013 at the keyboard into 012 at the UART.

The UART is programmed to generate the standard Teletype compatible format of a start bit, seven ASCII data bits, least significant first, then parity and stop bits. The key pressed signal from the keyboard unit is used as the data strobe to start transmission, and the transmitter end of character output of the UART is used to acknowledge completion of transmission. A 555 circuit is used to generate the clock. The clock should be adjusted to a 1760 Hz square wave; the circuit shown has about a 15% adjustment range for this purpose. The output of the UART is buffered by two inversions which protect the UART from excessive current drain. The buffered output in turn drives a relay through the quasi-Darlington coupled transistors. The relay used must be capable of switching the 60 mA current loop in times on the order of one millisecond. It must also be capable of sustained operation at high rates of change. If your junk box is not equipped with such a relay, other alternatives include use of an opto isolator and use of a high power interface circuit such as the 75451 driver chip.■

Watch Your

With Help, Of Course, From
Line of Compatible Plug-in

All our products are compatible with the 8800 — they just plug in to become an integral part of your system. Each card can be used in the widest range of applications, giving maximum versatility to your Altair. Our "no compromise" philosophy assures you the highest possible quality. We use a conservative thermal design to provide for long life and reliable operation. We've also put hysteresis bus receivers on noise prone high-speed inputs, giving you maximum noise immunity. And, finally, we cover it with a full six- month warranty.

Video Display Module

Your Altair already has the intelligence, so let us provide the display module. This is **not** a limited "TV Typewriter", but an ultra-high speed computer terminal built into your computer. The **VDM-1** generates sixteen 64-character lines from data stored in the 1K Byte on-card memory. Alphanumeric data is shown in a 7 x 9 dot matrix with a full 128 **upper and lower case** ASCII character set. The VDM-1 features EIA Video output for any standard video monitor. (Your TV set can be easily modified at your local television repair shop.) Multiple programmable cursors, automatic text scrolling and powerful text editing software are included **free.** Kit Price, $160.

A Versatile I/O Card

Just one **3P+S** card will fulfill the Input/Output needs of most 8800 users. There are two 8-bit parallel input and output ports with full handshaking logic. There is also a serial I/O using a UART with both teletype current loop and EIA RS-232 standard interfaces provided. The serial data rate can be set under software control between 35 and 9600 Baud. You can use your models 15, 28 or 33 TTY! This module gives you all the electronics you need to interface most peripheral devices with the 8800. And, should you decide to buy a 3P+S, we'll be happy to advise you on the best way to implement your system with our module. Kit Price, $125.

FOCAL!

Get a full 8080 implementation of 8K FOCAL* (including the game of Lunar Lander). It's now at your local Processor Technology dealer. Object tape is available for the copying charge only; complementary source listing available with minimum purchase.
(*® Digital Equipment Corp.)

An EPROM Module

Read Only Memories do not lose their stored data when power is removed. Thus, they have an advantage when used in stored program applications. Some ROM's, called EPROM's, are both erasable and reprogrammable. Our **2KRO** module will accept up to eight EPROM's, providing the user with up to 2048 eight bit words of non-volatile storage for monitor, executive, loader, and other programs. (We recommend the use of 1702A and MM5203 EPROM types. They are not included, but are readily available for reasonable prices on the industrial and surplus markets.) Kit Price, $50.

Frankenstein Emulation

Joe Murray
International Harvester, Solar Division
2200 Pacific Hwy
San Diego CA 92138

This is a let's get the ball rolling article. We now can analyze and build working models of at least portions of the human brain right in the home. Paper and pencil models of the brain develop naturally and almost without effort when we use real time digital design methods. The hardware and software mechanizations fall out naturally; then we just use the home computer lab to build what we have designed.

The Model

Let's follow the development of a crude and simple system engineer's model of the human "computing system." We look inwards, down into ourselves, and what is the first thing we see?

The Top Processor

This is the only unit that is really visible to the user. The Top CPU functions at the heart of the human control console. Here, our personality can sit down and use the entire human system to the limit of its capabilities. This visibility of only the input, output and manual control functions is typical of all computer systems from the hand calculator to the human brain; the rest of the system is invisible to the user and can only be deduced from what we see in the way of output response to input stimuli.

The Top Processor's Executive Program

Our personality uses the Top Processor as the system executive. The Top Processor is boss. Messages from the Top Processor set priorities for all the other elements in the human system. Exceptions to this rule are:

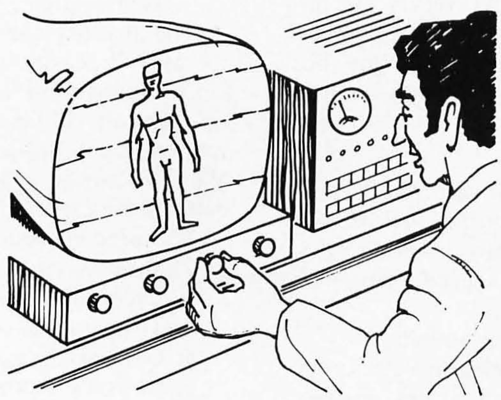
1. Emergency interrupts – a large set of emergency situations are fielded by faster, more powerful processors in subsystems.
2. Standard functions – built in executive programs in other processors manage tasks like circulation, digestion, etc., without bothering the Top Processor.

Top Processor Memory Allocations

The Top Processor has access to a limited scratch pad memory. However, this limited memory is used in an efficient mannner. The intersystem communication control programs can learn to transfer whole programs or portions of programs from the main memory banks to the Top Processor scratch pad memory. In a similar fashion small data sets can also be transferred. This is the familiar overlay manipulation (used in man made machines) that allows solution of complex problems in limited working memory by transfer to and from bulk storage units (as in magnetic disks and tapes).

The Top Processor's Use of Overlay

If the entire program and necessary data can all be stored in the scratch pad of the Top Processor, it simply executes the program on the data set and outputs the answer (example: 2 + 3 = 5). However, when the program and data set are too large to be loaded into the scratch pad memory, the program and data set are broken into sequential, related segments. The program is worked in segments and intermediate answers are stored. Final answers are output to our personality upon completion. Training can increase the power of this method; however, each of us has our own personal limit: For instance, I either lose some data or else lose my location in the program sequence. During the past few thousand years we humans have developed a host of



languages for communication. We also use these communication tools to extend the overlay method to more complex problems. We write down intermediate answers and manually track the execution of the program sequence. These languages include English, Polish, Spanish, arithmetic, algebra, Boolean logic, numbering systems, FORTRAN, PL/M (to name a few). The only limits on this extension of using the Top Processor in overlay fashion are:

1. Can we find the required data set?
2. Can we formulate the problem so as to allow a solution?
3. Do we have enough time?

This overlay use has become so powerful (with the help of the various languages) that we sometimes neglect a more ancient, natural, rapid and sometimes more powerful method to arrive at a solution. This method is to:

1. Develop the framework of the problem in the Top Processor.
2. Digest the available data within the framework of the problem.
3. Assign a high priority to the problem.
4. Send the above three items to faster, more powerful CPUs.
5. Sit back with a cup of coffee and wait for an answer.

When I follow this latter procedure, the return message is either:

1. The answer I seek.
2. The identification of missing data.
3. A question mark.
4. Garbage: (Garbage In implies Garbage Out – often abbreviated GIGO)

For answer 2, I go search for the missing data. For answer 3, I both search for missing data and review the framework of the problem for possible faults. For answer 4, I may use the garbage; I have carried some misconceptions for years.

Start the System Diagram

Let us summarize the Top Processor and place it in the system diagram. We've deduced by introspection that the Top Processor:

1. Is boss – The Top Processor is in direct communication with our personality and (with some exceptions) sets the priorities for the whole multiple processor system.
2. Has access to a small scratch pad memory.
3. Can fetch programs and data from the main memory bank.
4. Receives some body sensor data.
5. Communicates directly with other CPUs.

Figure 1 shows a pictorial summary of the system.

Data Bus Structure

The data bus structure is depicted in figure 1, using the normal multipath digital type of bus. However, empirical evidence implies a more complex communication system between elements of the human system. Just as the entire human system

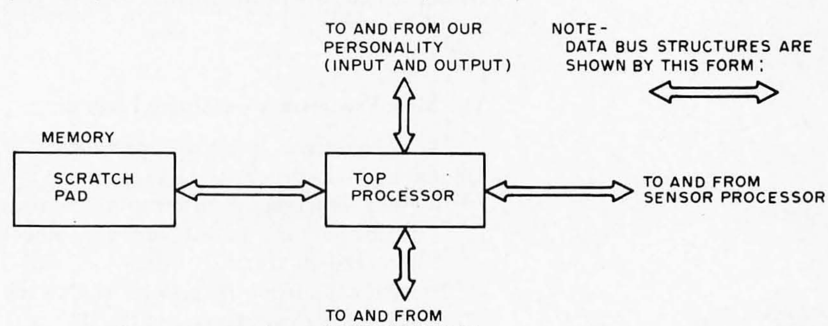


Figure 1: The Top Processor. Introspection starts at the immediately available evidence: We all have a Top Processor, our personality which controls most of our actions.

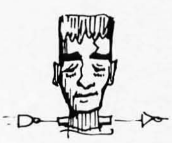
adapts to the use to which our personality puts it, this bus structure also adapts to how it is used. Witness the ease of recall on an often used phone number versus the difficulty in recall of a seldom used number. We might guess that somehow the bus structure is under adaptive software control.

The Main Processor

We now arrive at the general purpose powerhouse of the computing system. The Main Processor handles awe inspiring problems with unbelievable speed. We must postulate:

1. Elegantly simple programming.
2. Operation at a fast effective clock rate.
3. An outstandingly efficient internal executive program.
4. Access to the bulk of stored programs and data.
5. A complex priority interrupt system.
6. A multiple bus structure to the rest of the human system.

Main Processor Speed of Execution



The Main Processor is a very fast machine operating on elegant and simple programming. For instance, some of the muscle control programs must take only 20 to 50 milliseconds for completion of:

1. Input of data.
2. Computation on new data.
3. Output of control commands.
4. Cleanup for next computation period.

Navigation and guidance computation periods can be longer. However, they can not be much longer when we watch a small boy pick up a rock and knock a can off a fence post, all in the space of two to three seconds. Another awe inspiring feat is the performance of a businessman in his value judgment search as he keeps abreast of the rapid fire conflicts in the executive boardroom. The Main Processor seems to be an order of magnitude faster than the Top Processor (witness the increase in touch typing speed when the Top Processor gets out of the act).

The Main Processor's Executive Program

The executive program provides for scheduling Main Processor tasks that:

1. Field emergency interrupts such as avoidance of a fast moving object detected on visual sensors.
2. Take calls from the priority stack such as recognizing hunger and thirst.
3. Time share muscle control and evaluation of sensor data when both are active as in soccer game.
4. Regularly service body functions such as circulation, digestion, elimination, etc.
5. Start and stop background tasks such as meditation.

The quantity and variety of data used by the Main Processor in combination with the rapid response in answer to massive and conceptually difficult problems implies a very efficient software organization. The Main Processor must access tables that define the location of:

1. Stored life history data.
2. Muscle control programs.
3. Chemical control programs.
4. Temperature control programs.
5. Guidance programs.
6. Navigation programs.
7. Value judgment data.
8. System priority data.
9. System timing data.
10. Unused memory.

The Main Processor Decision Process

One of the most interesting functions of the Main Processor is to aid in the decision process we use when faced with alternate courses of action in response to events in the world around us. The evidence implies that the Main Processor takes formulation of the decision problem and the pertinent data from the Top Processor and Sensor Processors. These inputs are then heuristically compared to an immense value judgment table to generate a candidate decision. The candidate decision is sent to the Top Processor for further evaluation.

The Value Judgement Table

This table has a strong effect on the pathway we follow in life, from when we make the decision to start breathing until we are forced to stop breathing. How do entries appear in this table? Some entries must appear while we are within our mother. A new born infant makes the decision to start breathing or has an early death. Some entries come from trial and error experience. The young infant soon learns to cry just so mother will pick him up.

Some entries come from other people. The young child seeks his parents' approval, not their punishment. Another question: What can we know about entries in this table? We seem to know only recent, temporary residents such as priority on getting to the grocery store. The older, more permanent residents that have a continuing effect on our lives were either never known or long ago forgotten; yet there they sit, having a permanent effect on our success or failure in every endeavor (scares you, doesn't it?). Utility programs for determining the content

of this table and altering it can be implemented. This is sometimes accomplished through a verbal data link to an external Diagnostic Processor.

The Interrupt System

These interrupts are fielded in the Main Processor, and are used to re-direct effort, from meditation and decision processes to avoidance of a thrown rock or jumping away from a hot stove. The priority interrupt steers to the proper program without hesitation. Priority of the interrupts is used to decide which of several should be serviced.

The Main Processor Bus Structure

The Main Processor has a multitude of output and input data. Even in this crude, simple model, the resulting bus structure is quite complex. Let us add the Main Processor and connecting bus structure to produce the system diagram of figure 2.

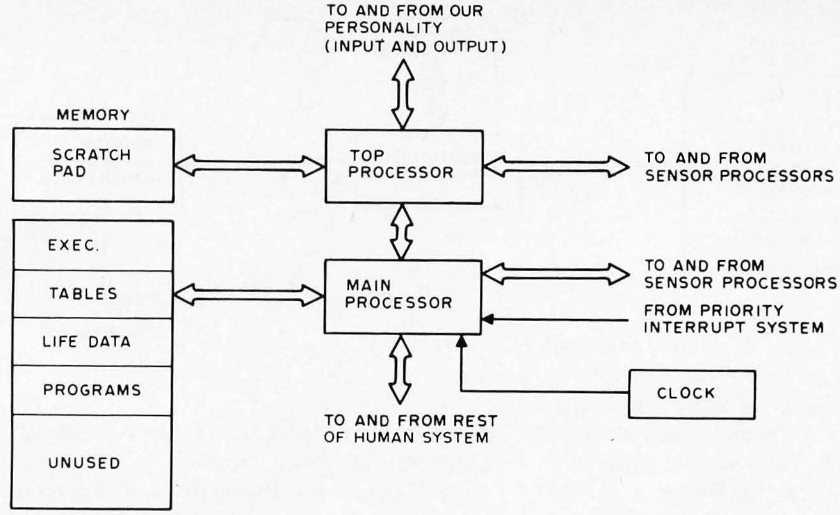


Figure 2: The Main Processor. Digging a bit deeper, we find that there is a lower level Main Processor which works cooperatively with the Top Processor to do a lot of the detail work in the system.

The Sensor Processors

The Sensor Processors are fast, special purpose units. Data is acquired from the eyes, ears, and a host of body sensors that continually look inside and outside the human system. The Sensor Processors for these devices execute programs that organize, compact and format this huge data flow for rapid and effective use by both the Top Processor and Main Processor. The introspective evidence implies:

1. A very fast clock rate.
2. Elegant and simple programs.
3. Access to a dedicated memory.
4. Existence of a buffer scratch pad memory for temporary storage of output data.
5. A very efficient executive program.
6. A complex input bus structure.

Intuitively one feels that sensor processing is not done by a single unit. Rather, an organization with a master processor and several dedicated slave processors would better fit the performance requirements. Each slave Sensor Processor could provide parallel service to the eyes, ears, etc. Figure 3 shows an addition to our system diagram to account for the master Sensor Processor and its slaves.

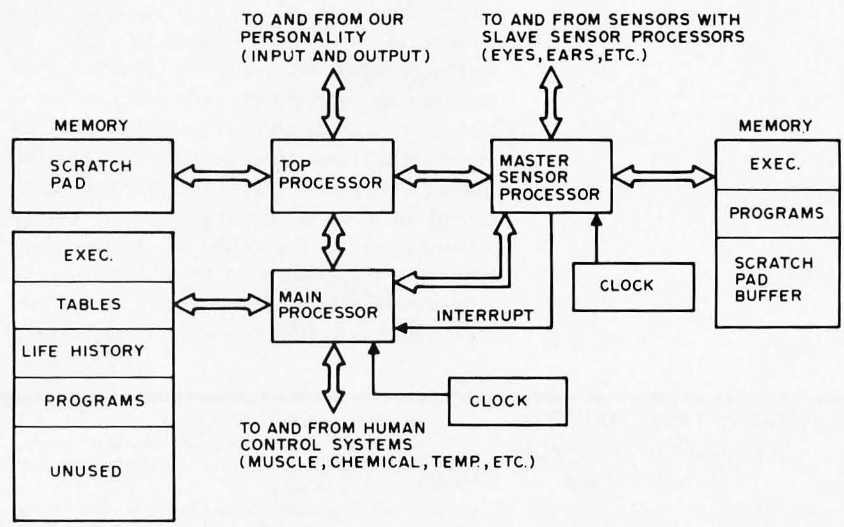


Figure 3: Adding the Sensor Processors to the System Concept. A system of Sensor Processors can be identified; they probably consist of a Master Sensor Processor with multiple Slave Sensor Processors dedicated to actual devices.

The Creative Process

All of us are creative; this is the way our personal human system adapts to the changing world around us. We create new machines, art objects, programs within our brain, communication languages, etc. The list is endless. Just how do we implement the creative process?

Let us postulate Random Pattern Generators for various creative tasks. The Sensor Processors can drive these generators with a supply of random combinations of data.

The Creativity Processor

The Creativity Processor uses the output of the Random Pattern Generators to build new logical structures or modify existing logical structures. These new structures are tested against requirements generated by the Top Processor. The value judgement process makes decisions that guide the Creativity Processor in continued improvement of the new design (in iterative, random fashion) until acceptance is obtained. The speed of

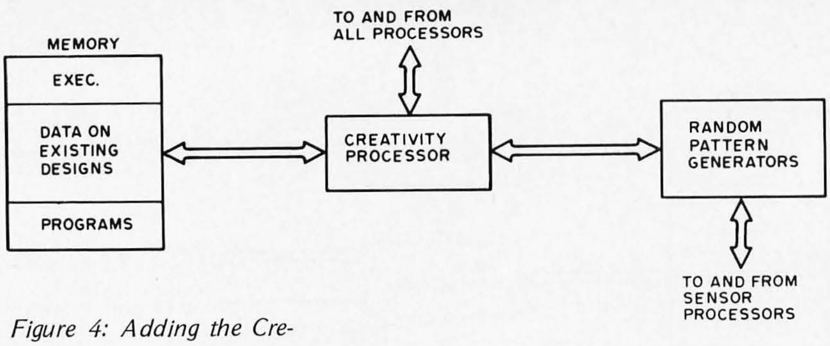


Figure 4: Adding the Creativity Processor to the System Concept. We must not forget about creativity. Interacting with the whole system is a matrix of creativity symbolized by the concept of Creativity Processor with its random pattern generation features.

the creative process has a heuristic design which improves with experience.

The Creativity Processor and interconnecting bus structure are shown in figure 4.

Data Set Manipulation

The data sets which are transferred throughout the system seem to be organized along the lines of various patterns (one picture is worth a thousand words). For instance, when we recognize someone, we seem to be recognizing some main features, not every detail that is available through close inspection. Visual data sets from the Sensor Processors seem to have been processed into some skeleton pattern before transmission to the other processors. Data from the ears seems to be stored in some logical thought structure pattern. I think out ideas both in picture and word format. Then, if my thinking was in picture format, I have trouble expressing my ideas verbally; whereas, if thought out in words beforehand, the expression of the ideas flows logically and clearly.

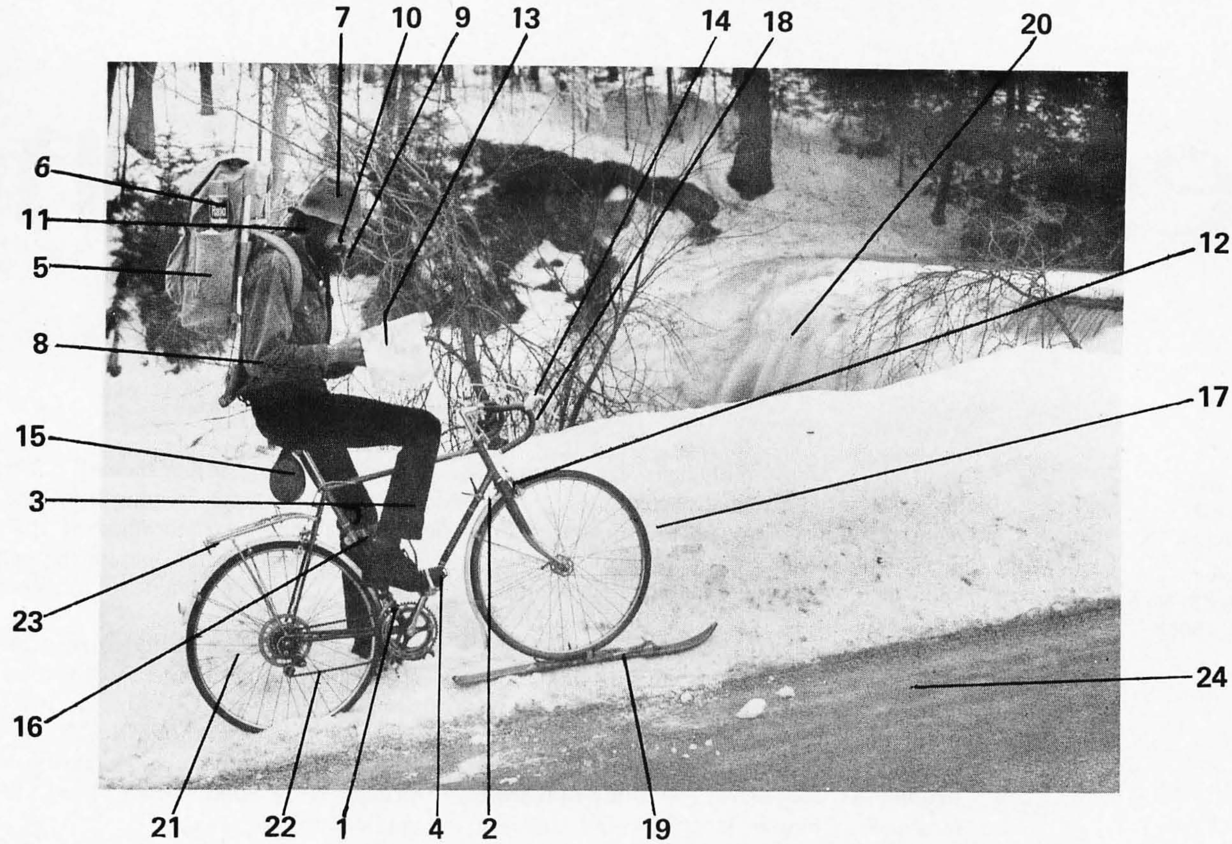
As in any control and guidance system, numerous feedback paths also exist. These were not detailed in this simple model.

Test the Model Validity

With a computer in the home laboratory, we have the means to test models of the human brain like this sketch. We can start with simple approximations and work our way up. Then, when our home brew computer system begins to perform like some portion of the human computing system, we have more than speculative evidence; we have truly come to know how that portion of the brain works. Also, some very useful hardware and software configurations may come out of the search.

Looking inward from the control console, we have followed the generation of a speculative, crude, simple, system engineer's model of the human computing system. Construction follows the line of man made, real time digital systems. In fact, one often suspects that designers of real time operating systems use very introspective models. This should make us optimistic that digital design tools are a natural and powerful approach to analysis of the human reasoning powers and control systems.■

Technology Update



BYTE always searches far and wide for the latest in the technology of computing systems. This month in the hills of New Hampshire, we discovered an example of computer technology in the form of the first practical Touring Machine, shown here complete with a unary relocatable based operator (in IBM OS PL/1 parlance).

For those individuals having less than a passing acquaintance with computer science, the Turing Machine is a famous mathematical construction first formulated some decades ago by Alan Mathison Turing, and which can be shown to be logically equivalent to any digital computer implementation. A Turing Machine is to computing what a Carnot Cycle is to thermodynamics. (The fact that this particular Touring Machine implementation looks like a CarNot Cycle is purely incidental.) But Turing machines have been notoriously impractical in terms of everyday computer usage until this new product rolled into town.

This newly released virtual Touring Machine, version 27 chain level 1, incorporates numerous state of the art features which make it one of the better examples of the form. These features include:

1. SHIFT (micro instruction).
2. 10 speed clock controls.
3. 2 phase clock drive.
4. clock conditioner.
5. LCS (large cookie store).
6. global debugging mechanism.
7. flying head with head crash padding.
8. access arm.
9. audio output peripheral.
10. visual input scanner.
11. audio input scanner.
12. local debyking mechanism.
13. relocatable memory mapping software.
14. HLT (halt instruction).
15. system maintenance package.
16. competing access lockout feature.
17. nomadic road interfaces.
18. tape.
19. SHIFT (macro instruction).
20. EXCP (executing channel program).
21. sectored disk drive.
22. transmission links.
23. unallocated stowage.
24. machine environment (circa January 30 1976).

Design an On Line

Robert R Wier
PO Box 9209
College Station TX 77840

James Brown
2518 Finley St No 636
Irving TX 75062

Machine or assembly language will most likely be used by many computer experimenters. While many professional programmers will swear by the use of assembly language, others, perhaps equal in number, will swear at it, preferring the use of high level languages. To those new to the field, these terms may seem confusing. It's really quite straightforward when one remembers that the language a machine uses differs considerably from the one used by the people. As one surveys a continuum from machine to human languages, the language most easily understood by the machine is a binary language; next on the continuum is assembly language with additional features that make it considerably easier to use, thus avoiding all night debug sessions, frazzled nerves, and 2 AM programming logic which hardly ever works, etc. For a good discussion on assemblers, see the October 1975 issue of BYTE. Easier yet for the programmer are languages such as BASIC, FORTRAN, PL/I, and ALGOL. These languages allow the problem to be stated and solved in terms better adapted to human understanding. Unfortunately, there are rather serious difficulties encountered when these high level languages are to be used on small systems. They require a compiler or interpreter to transform the problem from the high level language to machine language and more memory than is found in most small hobby systems. Therefore you'll probably be using assembly and machine language. After the program is written and loaded into the machine, experience has shown an astronomical probability against the program working correctly if it is more than two instructions large. Considerable time will probably be spent at the front control panel surveying the address and data lights, mumbling "I dontunnerstand" and "(expletive deleted) machine!". This can lead to terrific pains in the back and neck from bending over to look at the panel square in the face and operate the switches. This is commonly named "minicomputer neck."

How much nicer would it be to sit in a chair and do approximately the same thing using a Teletype or CRT display (CRT is an abbreviation for Cathode Ray Tube, essentially a TV picture tube. A television typewriter is a unit often used in this application).

There are several ways to use the control panel:

1: Executing a few instructions, then examining memory to see what the blinking machine is *really* doing, or
2: Inserting or changing data in memory, or
3: Displaying the contents of specific memory locations, or
4: Searching through memory for a specific bit string or number, if you prefer, or
5: Displaying and possibly changing the values in the CPU registers.

The authors had occasion to be working with a 16 bit/word minicomputer which mainly was used as a remote job entry terminal into a large computer. It could, however, function as a stand alone computer. Since an assembler was available, a number of assembly programs were written and debugged. When the machine was first delivered, a temporary control panel was provided. Since this was to be removed at some future date, the following technique was used to implement a DEBUG program using a CRT terminal to replace the control panel.

The basic idea is to develop a program that will take care of the functions outlined above and interface to the console terminal

Debugger

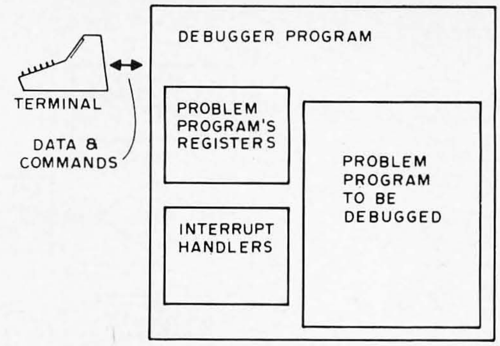


Figure 1: Logical arrangement of debugger.

and hopefully will protect itself from wild extremes of a program being debugged. This might be thought of as running a program within a program (figure 1). Hereafter, the program being debugged will be referred to as the problem program.

The debugger program must have provisions for a number of things. It has to handle the IO for the hardware and to converse with the human programmer. It has to keep track of the various status conditions of the program being debugged (the problem program). It must understand the input commands directing it to perform certain actions of the problem program. It must be transparent to the problem program so that when the final version is finished, the problem program may be loaded without the debugger, and still work.

In addition, the debugger should be small in size, and easy to implement to avoid the herculean task of debugging the debugger. (Although that's not strictly true. Once the IO and display portions were working, we used these to debug the rest of our debugger.)

The following commands are the results of our efforts to provide effective yet concise operations. In this list *adr* means a specific memory address, *val* a value, and *reg* a register.

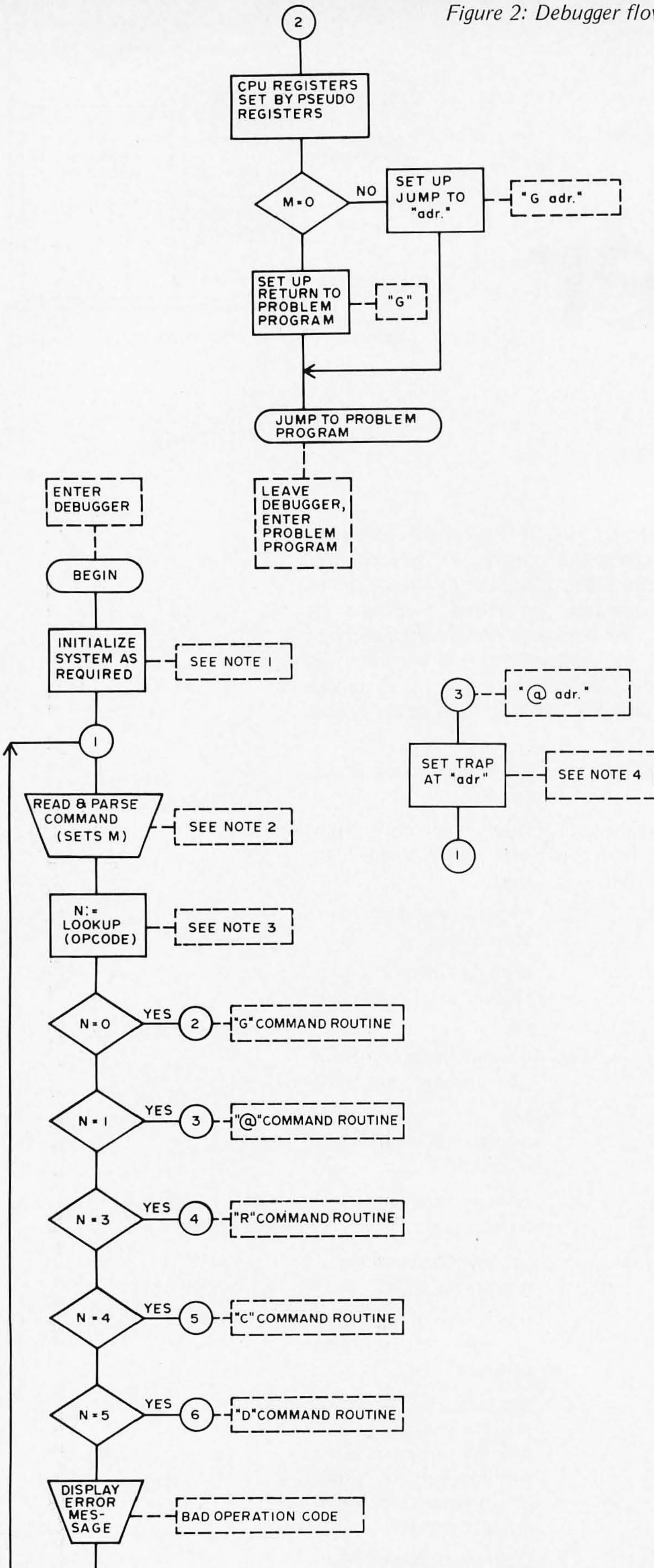
The final implementation including all the IO and interrupt handlers required 560 bytes, or about 256 instructions on the Lockheed SUE 1110 minicomputer. Figure 2 is an overview of the debugger logic flow. It is reasonably straightforward, except for the execute (G) instruction. Consider the debugger waiting for a programmer to enter a command. It just sits there wasting expensive electricity. As soon as you enter a command, the debugger checks it for vali-

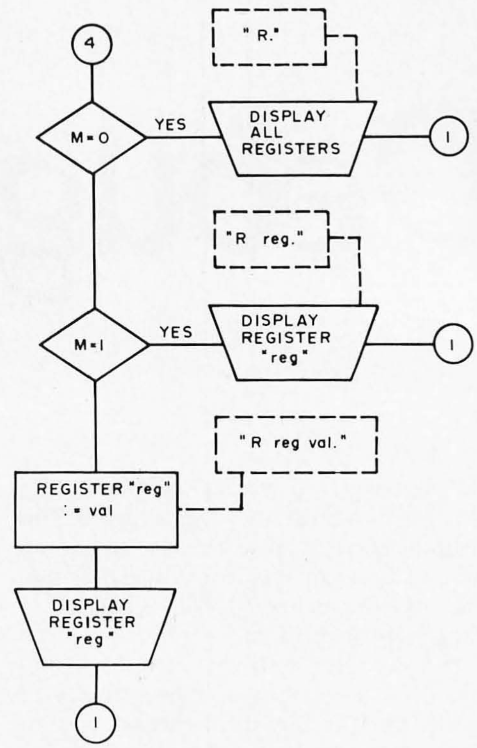
text continued on page 60

Table 1: DEBUGGER program commands. Each command consists of an operation code character, followed by from one to three operands (numbers) separated by blanks. The command line is completed by a period. In implementing the program, the computer should respond with a carriage return and line feed after finding the period.

C adr val.	changes memory at adr to val
C adrl adr2 val.	changes memory from adrl through adr2 to val
D adr.	displays memory contents at adr
D adrl adr2.	displays memory contents from adrl through adr2
D adrl adr2 val.	searches memory from adrl through adr2 for val
R.	displays the contents of all registers
R reg.	displays the contents of register reg
R reg val.	changes the contents of register reg to val
@ adr.	sets return to debugger at adr in problem program
G.	go, i.e., continue or start execution of problem program using contents of the problem program's program counter register
G adr.	start execution of problem program at adr

Figure 2: Debugger flowchart.





A TRAP EXAMPLE

Assume that the memory of the computer contains the following information.

Address	Content at this address
LOW	Beginning of the problem program region, a low address.
,	
,	
,	
LOC.W	Call DEBUGGER trap handler. This is a "trap."
LOC.W+D	Assuming a trap call of length D, this is the next instruction of the problem program after the trap.
,	
,	
LOC.X	Problem program ends.
LOC.X+1	Systems programming area begins (RAM, not ROM).
,	
,	
LOC.Y	DEBUGGER program starts.
,	
,	
LOC.Z	DEBUGGER's trap handler routine.
,	
,	
HIGH	End of systems programming area.

Note 1: The DEBUGGER program acts as a system monitor for your computer. Whenever the computer is restarted, the DEBUGGER is entered and will execute a power-on initialization sequence.

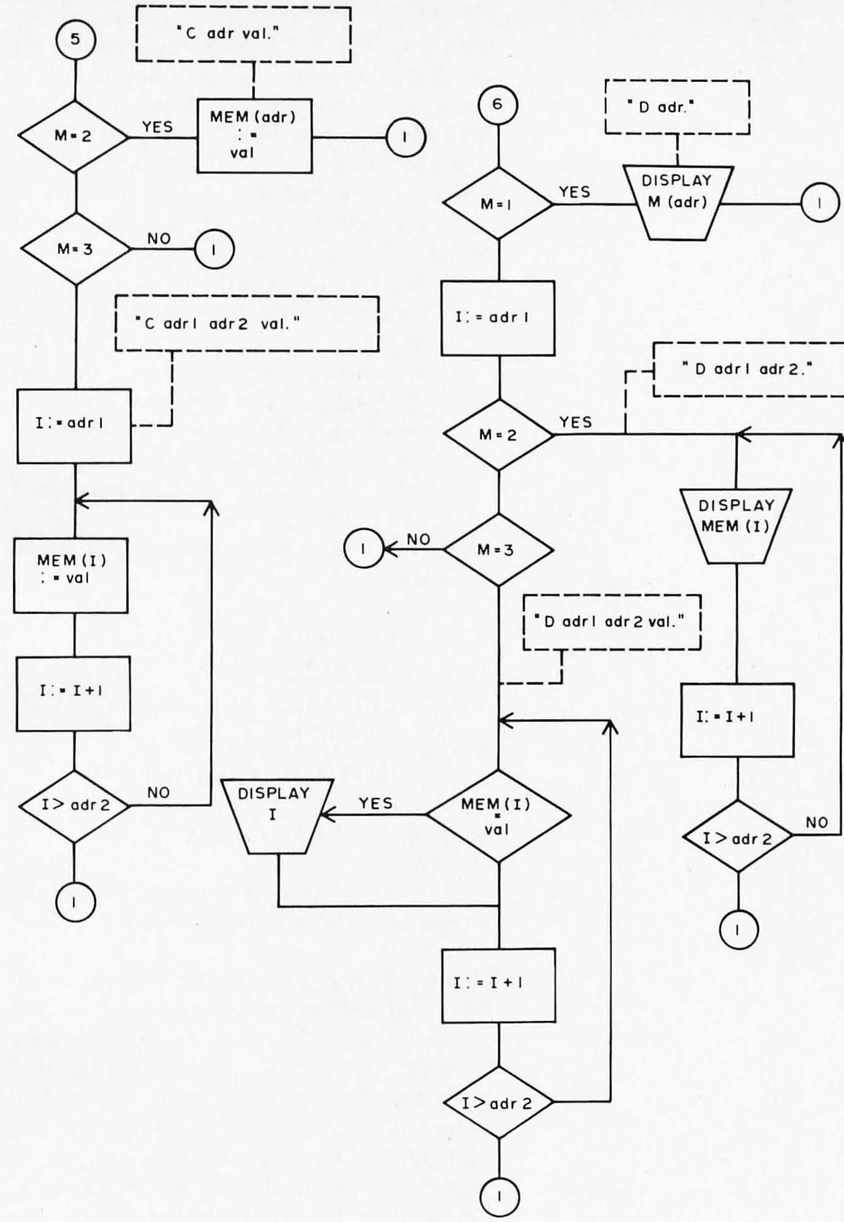
Note 2: The format of the command line and a list of all the variations on each command are found in table 1. The input routine should parse the command line by identifying the operation code and operands, stripping blanks, and counting the number of operands (M).

Note 3: The function LOOKUP is used to translate an input ASCII command character into a corresponding integer number. In the authors' system, this was accomplished by manipulating the bits of the ASCII character code; other schemes are possible.

Note 4: A trap is set by replacing the instruction at the trap address with a temporary alternate which causes a branch to the trap routine. The instruction used for this purpose in the authors' system was a "jump to subroutine." Depending upon the particular computer architecture, other instructions might be used, such as software interrupt, supervisor call, etc.

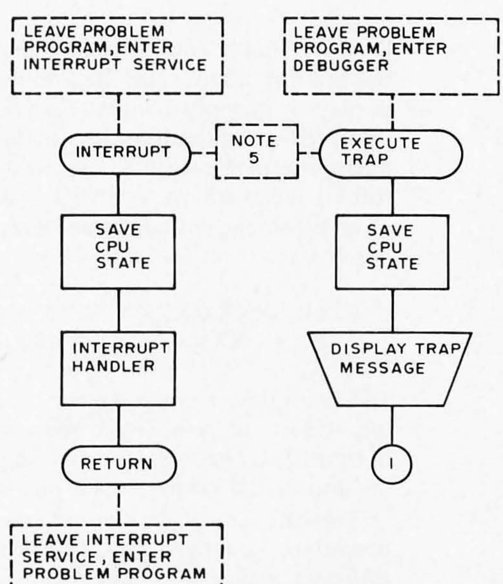
Note 5: Both trap instructions and interrupts require similar processing. One way to view the DEBUGGER program is as a large interrupt handler which is entered upon system restart, execution of a trap, or end of a problem program's execution.

Note 6: Command formats from table 1 are shown in quotes within comment boxes in this flow chart.



Assuming a stack oriented machine in which the state information is stored in the stack, the following sequence occurs in a typical case.

1. The user enters a program. After entering it, he decides to place a trap at location LOC.W in memory with the "@" command.
2. The problem program begins execution after a "G LOW." command. It reaches the trap at LOC.X and executes the subroutine call.
3. The subroutine call saves the address of the next instruction (at a minimum) and branches to the trap handler at LOC.Z. The trap handler continues the state saving process so that the machine's stack contains complete CPU state information.
4. The trap handler displays a trap message containing information on the address and register content of the machine at the time of the trap.
5. The trap handler passes control back to the DEBUGGER's command line interpreter.



text continued from page 57

dity, and if it is a legitimate command the various parameters are read and stored in memory to be accessed when necessary. Now the debugger looks at the part of the command line which tells it what to do (known as the opcode). Assuming that you are using ASCII, here is a sneaky way of determining which routine to go to.

1: Add 9 to the ASCII character,
2: Logically AND the opcode character with a 7,
3: Assuming the given opcodes (C,D,R,@,G), you now have a numerical index which you may use to test or use in a jump table to go to the proper code which accomplishes the desired function.

EXAMPLE: Suppose you have an ASCII 'R'; in binary this is:

```
0101 0010 - 'R'
0000 1001 - add 9
---------
0101 1011 - AND result
0000 0111 - with 7
---------
0000 0011 - final result is '3'
```

using this method then: G=0, @=1, R=3, C=4, D=5.

Now we offer a few comments on the various procedures shown in figure 2.

Change: This is perhaps the simplest of all the commands to implement. Using the last parameter supplied, step through memory from the first address zapping each location with the desired value until the ending address is reached (note: for a single address, adr1=adr2). Though not necessary, it is highly recommended to check the addresses for validity to avoid clobbering the debugger.

Display: Simply step through memory from the starting address to the ending address displaying memory contents as you go. We displayed in hexadecimal notation. You might alternately wish to use octal or (God forbid) binary. Since our CRT was capable of an 80 character line, we put 8 groups of 4 hex characters on each line:

```
LLLL: XXXX XXXX XXXX XXXX
      XXXX XXXX XXXX XXXX
```

The first number is the memory location of the lowest address displayed on the line (leftmost). Using this, it is easy to glance at the display and see patterns in memory.

For the search option of the display operation, you need only to print out the *addresses* where a compare was successful. You should be able to remember what you are looking for. When the search option is used, a flag is set which somewhat modifies the display such as:

```
: LLLL LLLL LLLL LLLL
  LLLL LLLL LLLL LLLL
```

where the L's are the memory addresses containing the argument.

Some commercial variants of the search operation allow you to look for certain bit patterns within words by masking out don't care bits; however, this is no small task to program for a feature of somewhat limited usefulness.

Register: Here you have three alternatives determined, once again, by the number of operands (i.e., how many parameters you specify with a particular opcode). No operands are used to indicate the display of all register contents. If one operand is present, then the content of that register only is to be displayed. Two operands indicate the contents of the specified register are to be changed to the given value.

Please note that these registers are really fixed memory locations, set aside inside the debugger (i.e., pseudo registers). These values are typically loaded into the CPU registers by the G command. Most CPUs have one or more general registers plus a program counter (i.e., the address of the next executable instruction), and a collection of indicators commonly referred to as status flags or sometimes as status registers. For our implementation we had seven general registers numbered (cleverly) one through seven. Register number zero was the program counter and register number eight was the status register (note: All registers were 16 bits large). Thus we only had to enter a single digit, zero through eight, to reference any register. On most micro or minicomputers, alphabetic type designators are used to reference registers, but with much luck a similar trick used to simplify opcode determination may be used.

GO and SET TRAP: This section is the most machine dependent implementation which requires very careful planning. The object here is to put the problem program into execution, and eventually have control returned *gracefully* to the debugger. The point where execution is to end and control to return to the debugger is called a breakpoint or trap.

Constructing a trap is not too difficult. The simplest method is to insert in the problem program an unconditional branch back to the debugger. A serious drawback of

Figure 3: How to set traps in the problem program (see text).

MEMORY MAP

Address	Contents
LO	Problem program starts
.	
.	
.	
W	Call debugger trap handler at address Z
W+1	Problem program continues
.	
.	
.	
X	Problem program ends
X+1	Stack starts
.	
.	
.	
X+n	Stack ends
Y	Debugger program starts
.	
.	
Z	Trap handler of debugger program
.	
.	
HI	Debugger program ends

ALGORITHM

The stack of n elements is located at address X, the debugger program at address Y, and the trap handler at Z. The following steps are executed:

1: The problem program executes a trap at location W, i.e., a subroutine call to the trap handler.
2: The subroutine call saves the address W+1. return address W+1 in the stack, e.g., in X+3
3: The trap handler at address Z is executed.
4: The trap handler fetches the return address W+1 from the stack (in this example X+3), reduces the stack by one element, and displays the address W+1.

this is that the location from which the branch occurred will be unknown. The solution is to use an unconditional subroutine call to the debugger. A call instruction places a return address somewhere, depending on the machine, and then branches to the location specified in the instruction. With this it is a simple matter to retrieve this return address as the program counter for the 'G.' option of the GO statement (figure 3). Our computer had fixed locations in which routine addresses could be placed, such that if certain types of interrupts occurred the return address was saved and a branch taken using the address at that location (vectored interrupts). One such interrupt was a "bad" instruction interrupt, hence the setting of program traps consisted of moving an illegal instruction to the location a trap was to occur.

The GO command should set the pseudo program counter if an operand is present, then load all general registers. The last two registers loaded are the status flags and the program counter (which would be identical to a branch). Typically a branch using the contents of the pseudo program counter would be used (note: Branches usually do not set or reset status flags).

In conjunction with the preceding, there should be a phantom routine which is the target for all traps. Its job is to save all registers and status before the debugger main routine uses them into the pseudo register area. It is suggested to display the program counter and the fact that a trap occurred, such as:

@ interrupt address

There is a dandy reason for this. If multiple traps exist, it is handy to know which trap was encountered. Additionally, since the trap itself may clobber one or more memory locations in the problem program, to remove a trap one must change these trap instructions back to the original contents (typically from the original assembly listings). In an earlier version of the debugger we allowed only one trap per execution and saved the good code from the trap location. When the trap occurred, we then restored the good code at that location. However, a serious drawback, of course, was that it isn't always known what branches will be taken between the G and @ instructions, and it was highly probable that the trap would be bypassed entirely. Thus in our present debugger we allow multiple traps but do not restore the previous code when a trap occurs.

Execute Instruction Considerations: If you happen to get tied up in an endless loop, you'll have to manually force a return to the debugger. This could be accomplished in several ways. You could physically reset the machine from the control panel (assuming you have one), and enter the debugger starting address. Or you could have previously set up an interrupt structure which would respond to some outside stimulus (such as an escape from the keyboard, or a special control panel switch) which would accomplish a branch to DEBUG. Some thought was given to simply kicking the power supply, initiating a power fail interrupt, but this was later discarded.

If you make extensive use of interrupts in the debugger (which is not really necessary) then you'll have to debug your problem program's interrupts separately. Otherwise the problem program's interrupts and the debugger's interrupts will be working at cross purposes.

Should you place the breakpoint address in a branch of a conditional statement that doesn't happen to be executed, then the program will just skip around your break-

point. Or worse, placing the trap instruction as the operand of a multiword instruction could be distressing. The obvious solution to the first problem is the placing of multiple traps, so the problem program could not escape from the debugger regardless of the flow of control. The latter had no fool proof solution except exercising a little caution as to trap locations.

Some commercially available debuggers are really monitors that check the program counter every time a step is executed (interpreters). With a little thought it is apparent that this would involve considerably more programming than we've discussed here. Our debugger just allows you to set up the initial conditions and then "let fly," while the alternative is to have the debugger arrange every instruction which has the advantage of a more fool proof operation. But, it suffers from program complexity and a tendency toward slow execution which is critical in some IO operations.

The debugger ideally should be immune to anything which the problem program might try to do to it. This suggests the use of ROM (Read Only Memory). After you have the debugger working to your satisfaction, just place the debugger somewhere in your memory address space where you'll probably

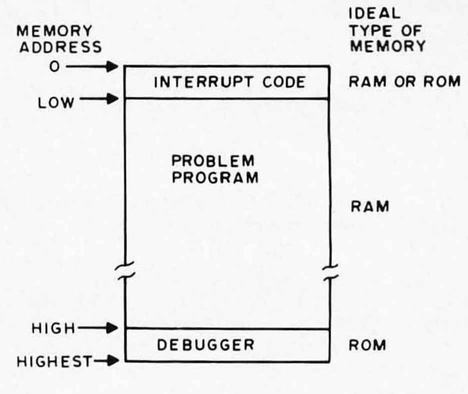


Figure 4: Physical arrangement of debugger in memory.

never need to move it. Usually this is in high memory. Since the debugger needs a small amount of RAM (Random Access Memory) in order to save the problem program's registers between G instructions, it cannot be made completely invulnerable. If the problem program happens to move garbage into the interrupt vectors, there is no telling what will happen on the next interrupt. But this is better than having the debugger completely in RAM. As a practical note, however, we found that there were not too many occasions when the problem program zapped the debugger if it was in RAM (figure 4).

If you want to get really fancy, you could include in the debugger an option to perform loading functions, such as retrieving a program off cassette tape. Assuming the debugger is in ROM you would never have to toggle in a bootstrap loader again, which is undoubtedly one of the worst aspects of small systems. Of course if you do not wish to get that fancy, you may still enter the loader via the debugger, which is certainly easier than using the front switches.

All in all, we've found that a good online debugger program is worth its weight in ROM. It will remove some of the worst aggravations of using small systems, and you'll learn a lot about logical flow of control, hardware software interfacing, and modularity of programming.

So let's get in there and STAMP OUT THOSE BUGS!■

Source listings of the debugger are available for the SUE 1110. Send one dollar to cover duplication and postage to Robert R Wier.

A version utilizing Intel's 8080 CPU chip is in the works, and when available a note will appear in BYTE.

HEWLETT PACKARD

H.P. IS BACK!



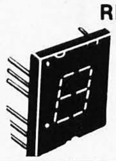
No Rejects or Retests here — You Want Quality? You have it with this popular 0.3" L.E.D. by Hewlett Packard. Fits Standard I.C. Socket. Right hand DP.
5082-7740 — Common Cathode $1.25
6 for $6.00

FAIRCHILD BIG LED READOUTS

A big .50 inch easy to read character. Now available in either common anode or common cathode. Take your pick. Super low current drain, only 5 MA per segment typical.

FND - 510 Common Anode
FND - 503 Common Cathode
YOUR CHOICE $2 ea.

DO YOU NEED A LARGE COMMON ANODE READOUT AT A FANTASTIC PRICE?



S. D. presents the MAN-64 by Monsanto - .40 inch character. All LED construction - not reflective bar type, fits 14 pin DIP. Brand new and factory prime. Left D.P.
$1.50 ea. 6 for $7.50

12 DIGIT R... — $1.75

SOLD OUT

Mfg. by National Electronics. B... ze. Gas discharge type, for use in desk top calculators, fre... clocks, etc. #NDP 1253-12. Orginally cost a calculator mfg. $16.77 ea. in quantity. *S.D. Price $1.75 ea.*

HIGH V. TRANSFORMER for above readout. Primary 117V, two secondaries: 24 VAC 500 MA 175 VAC 100 MA. $1.25 ea.

2102 1K RAM's - 8 FOR $12.95

New units ———— We bought a load on a super deal, hence this fantastic price.
Units tested for 500NS Speed.

INTEL 1702A 2K ERASEABLE PROM'S $6.95

We tell it like it is. We could have said these were factory new, but here is the straight scoop. We bought a load of new computer gear that contained a quantity of 1702A's in sockets. We carefully removed the parts, verified their quality, and are offering them on one heck of a deal. First come, first served. Satisfaction guaranteed.

SIGNETICS 1K P-ROM

PROMS!

82S129. 256 X 4. Bipolar, much faster than MOS devices. 50 NS. Tri-state outputs. TTL compatible. Field programable, and features on chip address decoding. Perfect for microprogramming applications. 16 pin DIP. With specs. $2.95 ea.



$2.95

PROFESSIONAL QUALITY STEREO HEADPHONES

Here is the kind of super deal that S. D. is famous for. Treat your ears to a super sound at a super price. Soft padded ear cushions, lightweight, fully adjustable headband, long coiled cord, wide response. **$6**
NEW IN ORIGINAL FACTORY BOXES

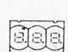
INSTRUMENT KNOBS

Black with brushed aluminum insert. Medium size, very attractive style. **SPECIAL 5 FOR $1**

MV-50 TYPE LED's

by LITRONIX
10 for $1
Factory Prime!

3 DIGIT LED ARRAY — 75c



by LITRONIX
DL33MMB. 3 MAN-3 Size Readouts in one package. These are factory prime, not retested rejects as sold by others. compare this price! 75c 3 for $2.

SALE ON CUT LEAD SEMICONDUCTORS

Leads were cut for PCB insertion. Still very useable.

1N914/1N4148 .... 100/$2
1N4002 1 Amp 100 PIV .... 40/$1
1N4745A 16V 1W Zener .... 20/$1
EN2222 NPN Transistor .... 25/$1
EN2907 PNP Transistor .... 25/$1
2N3904 NPN Driver Xstr. .... 25/$1
2N3392 GE Pre-amp Xstr. .... 25!$1
C103Y SCR. 800MA. 60V. .... 10/$1

ALL NEW. UNUSED. SOME ARE HOUSE #

NATIONAL MM5375AA ALARM CLOCK CHIP

Second generation alarm chip. Six digits, internally generates alarm tone, snooze, power failure indicator, etc. Very easy to use. Outperforms older types like 5316, 5370, etc. Perfect for use with our HP 5082-7740 LED readouts.
S.D. SPECIAL PRICE: $3.50
WITH DATA

MOSTEK MK50380 DIRECT DRIVE ALARM CHIP

Drives LED readouts direct, eliminates transistor or IC interface circuitry. 4 digit output is non-multiplexed to eliminate RFI in clock radio applications. Has sleep and snooze features. 40 pin DIP. Makes a JUMBO alarm clock by using our FND 503 readouts. We show you how.
INTRODUCTORY PRICE: $3.95

SLIDE SWITCH ASSORTMENT



Our best seller. Includes miniature and standard sizes, single and multi-position units. All new, first quality, name brand switches. Try one package and you'll reorder more. *Special — 12 for $1* (Assortment)

DISC CAP ASSORTMENT

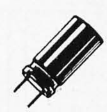


PC leads. At least 10 different values. Includes .001, .01, .05, plus other standard values.
60 FOR $1

UPRIGHT ELECTROLYTIC CAPS

47 mfd 35 V-10/$1 68 mfd 25V-8/$1
Brand new by Sprague. PC leads.

1000 MFD FILTER CAPS



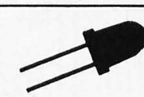
Rated 35 WVDC. Upright style with P.C. leads. Most popular value for hobbyists. Compare at up to $1.19 each from franchise type electronic parts stores. *S.D. Special 4 for $1*

RESISTOR ASSORTMENT



1/4 W 5% and 10%. PC leads.
A good mix of values. 200/$2

LARGE SIZE LED LAMPS



Similar to MV5024. Prime factory tested units. We include plastic mounting clips which are very hard to come by.
Special 5 for $1

VERNIER DIAL

From a close out of metal detector manufacturer. ½ Turn, 8 to 1 ratio. Internal stops easily removed to make unit multi-turn.
LIMITED QUANTITY - $.99 EACH

741C OP AMPS



Prime, factory tested and marked. Full spec on all parameters. Not re-tested, functional only, units as sold by others.
741 CH -TO-5 8 Lead Metal Can . . . 3/$1
741CV - 8 Lead Mini Dip . . . . 4/$1

DUAL 741C (5558) OP AMPS

Mini dip. New house numbered units by RAYTHEON.
4 FOR $1

FET'S BY TEXAS INSTRUMENTS — SPECIAL 5 for $1

#TIS-75 but with an internal house number. TO-92 plastic case. N. Channel, Junction type FET.

Signetics, TRI-State Hex Buffer. MOS and TTL interface to TRI-State Logic.
Special: $1 **8T97B**

TRANSISTORS
2N3904 - NPN
2N3906 - PNP
8 for $1

TI POWER TRANSISTORS

TIP29 NPN Silicon. TO-220. 4/$1

CT-5005

WITH FOUR-FUNCTION MEMORY!

CALCULATOR CHIP
By CalTex. 12 Digits.
With Specs. $1.49
Factory prime. Not retested

We do not sell junk. Money back guarantee on every item. No C.O.D. Texas Res. add 5% tax. Postage rates went up 30%! Please add 5% of your total order to help cover shipping.

S. D. SALES CO.

P.O. BOX 28810
DALLAS, TEXAS 75228

Microprocessor Update: Texas Instruments TMS9900

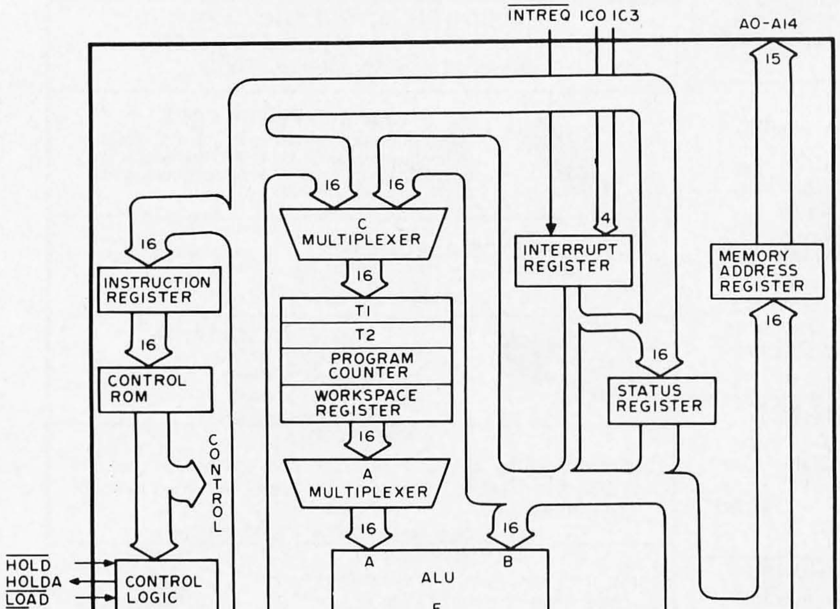


Figure 1: Internal Block Diagram of the TMS9900 Processor. The internal organization of the processor uses a microprogrammed approach. The control ROM which is part of the chip is used to store detailed sequences of internal operations within an instruction cycle. Programmer accessible memory on the chip is limited to the program counter, work space register and status register. From a programmer's point of view, nearly all operations concern memory which is addressed by bit lines A0 to A14 with data transfers over the bus lines D0 to D15.

Robert Baker
34 White Pine Dr
Littleton MA 01460

The TMS9900 microprocessor is a single chip, 16 bit central processing unit that requires power supplies of +5V, -5V and +12V, as well as a four phase, 3 MHz clock. Fast interrupt response and programming flexibility is provided by the implementation of a unique memory to memory architecture with multiple register files resident in memory. The instruction set includes hardware multiply and divide instructions making the TMS9900 comparable with many minicomputers. A compatible set of MOS and TTL memory and logic function support circuits together with separate data and address buses help simplify the system design. For industrial users, the TMS9900 system is fully supported by both software and a prototyping system. Figure 1 shows a functional block diagram of the TMS9900 CPU chip while figure 2 shows the actual pin assignments of the 64 pin dual in line package. Table 1 gives a detailed description of each pin, grouped by functions for easy reference. Photo 1 illustrates the unique large package of this processor.

Memory

The maximum addressable memory space is 65,536 bytes or 32,768 words. Each 16 bit memory word also defines two bytes of 8 bits. The word and byte formats are shown in figure 3. All memory word locations are even addresses and either the even or the odd byte may be addressed by byte oriented instructions.

The first 32 words of memory are used for interrupt trap vectors as shown in the

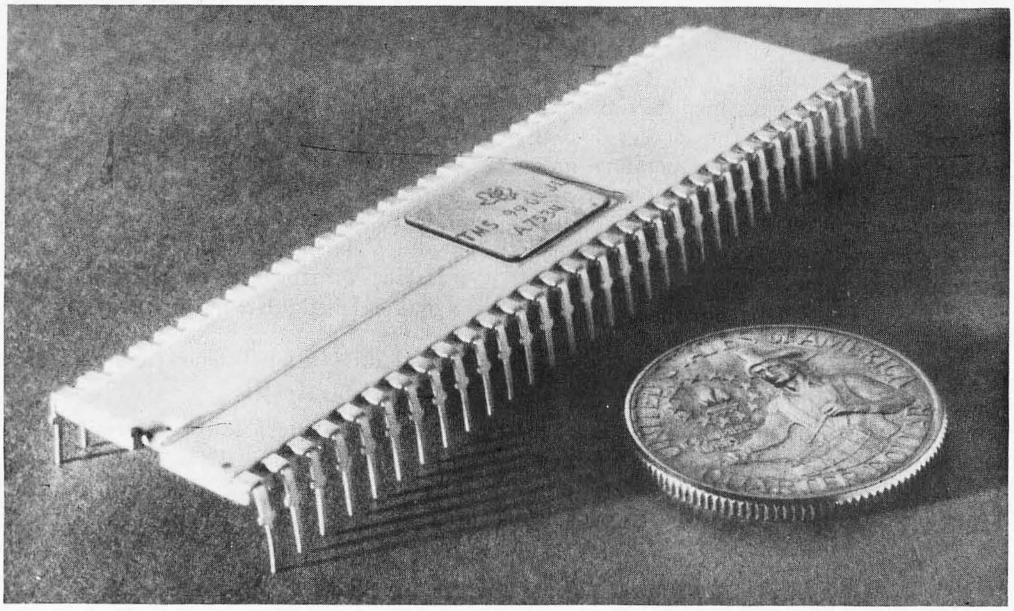


Photo 1: The TMS9900 processor is a 16 bit CPU with an advanced instruction set comparable to many minicomputer designs. It is packaged in this unique 64 pin dual in line package. The large number of pins available to the designers allowed the use of a fully parallel 16 bit data bus and separate 15 bit address bus. No address latching or multiplexing is required.

memory map of figure 4. The next block of 32 memory words is used by the extended operation (XOP) instruction for trap vectors while the last two memory words (addresses FFFC and FFFE) are used for the trap vector of the LOAD signal. The remaining memory is available for programs, data, and workspace registers as desired.

Registers

The first of three internal registers accessible to the user is the program counter (PC) which contains the address of the instruction following the current instruction being executed. This register is automatically incremented by two after being referenced by the processor to fetch the next instruction. The status register (ST) records the present status of the processor using the bits as defined in table 2. The workspace pointer (WP) contains the starting address of the currently active set of workspace registers.

The TMS9900 has an advanced memory to memory architecture utilizing blocks of memory designated as workspaces in place of internal hardware registers. A workspace register file uses 16 contiguous memory words of the general random access memory area. Each workspace register is a general purpose register available to the programmer as a temporary storage register, operand register, accumulator, address register, or index register. In the hardware of the chip, individual registers are addressed by adding the specific register number to the contents of the workspace pointer during instruction execution. Several of the workspace registers have fixed uses as part of subroutine and interrupt linkage conventions.

This workspace concept allows fast and easy process swapping or context switches (as in the case of interrupts) by exchanging only the PC, ST, and WP. No other register saving is required since each process retains its own set of general registers which are only used by its program.

Interrupts

Of the 16 available interrupt levels, Level 0 is reserved for the RESET function while all other levels may be used for any external devices. Level 0 is the highest and level 15 is

Figure 3: Memory Data Formats. The TMS9900 is oriented to a 16 bit word length; however, addressing to the byte level is assumed. The processor provides the programmer with 16 address bits for an address space size of 65,536 bytes; actual data transfers are 16 bits (two bytes) wide so the low order bit is never sent out to the memory. TI documentation uses the bit numbering conventions shown here. As is the case with most microcomputers, two's complement arithmetic is used for integer operations. The high order bit of the word or byte is treated as the algebraic sign of the number.

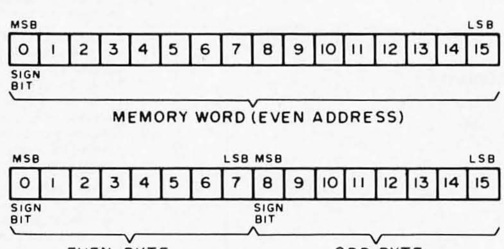
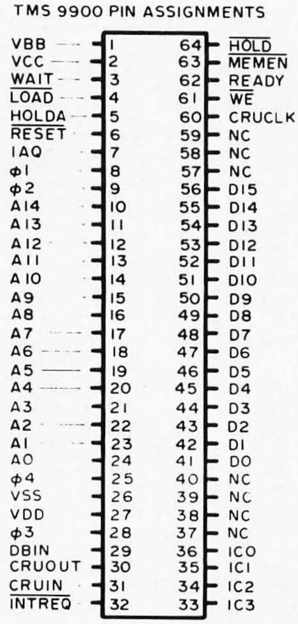


Figure 2: TMS9900 Pin Assignments. The 64 pin package is probably the largest commercially available integrated circuit in regular production at the time of this writing.

TMS 9900 PIN ASSIGNMENTS



NC-NO CONNECTION

POWER SYMBOL	VOLTAGES PIN	VALUE
VBB	1	-5V
VCC	2	+5V
VSS	26	GND (OV)
VDD	27	+12V

The TMS9900 is the first generally available single chip processor to provide hardware multiply and divide instructions.

Information and diagrams in this article are based upon the TMS9900 Microprocessor Data Manual, *courtesy of Texas Instruments Inc.*

the lowest priority level. Several devices may share any external level (1 to 15) as desired.

The interrupt code (IC0 to IC3) is continuously compared with the interrupt mask (ST bits 12 to 15). The processor will recognize an interrupt if the pending interrupt level is less than or equal to the interrupt mask level. Following completion of the current instruction an acknowledged interrupt will cause a context switch. The new WP and PC will be obtained from the interrupt vector locations and the previous WP, PC, and ST are stored in workspace registers 13, 14, and 15, respectively, of the new workspace. The interrupt mask is set to one less than the level being serviced (except for level 0 which loads 0) to allow only higher priority interrupts to be recognized. Interrupts are inhibited until the first instruction of the service routine has been executed to preserve program linkage in case of higher priority interrupts. All pending interrupt requests should remain active until recognized and each service routine should terminate with the return instruction (RTWP) to restore the original parameters. Figure 5 illustrates a typical TTL interrupt interface with priority encoding while table 3 gives detailed interrupt level data.

Input and Output: The Communications Register Unit (CRU)

Up to 4096 directly addressable input bits and 4096 directly addressable output bits are provided by the communications register unit (CRU) which is a direct, com-

Table 1: Detailed Description of TMS9900 Pins. This table lists the various pins of the TMS9900 along with an explanation of the purpose and use of the signals.

SYMBOL	PIN	IO	DESCRIPTION
			ADDRESS BUS
A0 (MSB)	24	OUT	A0 through A14 comprise the address bus. This 3 state bus provides the memory address vector to the external memory system when $\overline{MEMEN}$ is active and IO-bit addresses and external instruction addresses to the IO system when $\overline{MEMEN}$ is inactive. The address bus assumes the high impendance state when HOLDA is active.
A1	23	OUT	
A2	22	OUT	
A3	21	OUT	
A4	20	OUT	
A5	19	OUT	
A6	18	OUT	
A7	17	OUT	
A8	16	OUT	
A9	15	OUT	
A10	14	OUT	
A11	13	OUT	
A12	12	OUT	
A13	11	OUT	
A14 (LSB)	10	OUT	
			DATA BUS
D0 (MSB)	41	IO	D0 through D15 comprise the bidirectional 3 state data bus. This bus transfers memory data to (when writing) and from (when reading) the external memory system when $\overline{MEMEN}$ is active. The data bus assumes the high impedance state when HOLDA is active.
D1	42	IO	
D2	43	IO	
D3	44	IO	
D4	45	IO	
D5	46	IO	
D6	47	IO	
D7	48	IO	
D8	49	IO	
D9	50	IO	
D10	51	IO	
D11	52	IO	
D12	53	IO	
D13	54	IO	
D14	55	IO	
D15 (LSB)	56	IO	
			POWER SUPPLIES
V_{BB}	1		Supply voltage (-5V NOM)
V_{CC}	2		Supply voltage (5V NOM)
V_{DD}	27		Supply voltage (12V NOM)
V_{SS}	26		Ground reference
			CLOCKS
01	8	IN	Phase 1 clock
02	9	IN	Phase 2 clock
03	28	IN	Phase 3 clock
04	25	IN	Phase 4 clock
			BUS CONTROL
DBIN	29	OUT	Data bus in. When active (high), DBIN indicates that the TMS9900 has disabled its output buffers to allow the memory to place memory-read data on the data bus during $\overline{MEMEN}$. DBIN remains low in all other cases except when HOLDA is active.
$\overline{MEMEN}$	63	OUT	Memory enable. When active (low), $\overline{MEMEN}$ indicates that the address bus contains a memory address.
$\overline{WE}$	61	OUT	Write enable when active (low), $\overline{WE}$ indicates that memory-write data is available from the TMS9900 to be written into memory.
CRUCLK	60	OUT	CRU clock. When active (high), CRUCLK indicates that external interface logic should sample the output data on CRUOUT or should decode external instructions on A0 through A2.
CRUIN	31	IN	CRU data in. CRUIN, normally driven by 3 state or open collector devices, receives input data from external interface logic. When the processor executes a STCR or TB instruction, it samples CRUIN for the level of the CRU input bit specified by the address bus (A3 through A14).
CRUOUT	30	OUT	CRU data out. Serial IO data appears on the CRUOUT line when an LOCR, SBZ, or SBO instruction is executed. The data on CRUOUT should be sampled by external IO interface logic when CRUCLK goes active (high). With software drivers the CRU serial interface can be used in place of a UART or similar conversion IC.
			INTERRUPT CONTROL
$\overline{INTREQ}$	32	IN	Interrupt request. When active (low), $\overline{INTREQ}$ indicates that an external interrupt is requested. If $\overline{INTREQ}$ is active, the processor loads the data on the interrupt code input lines IC0 through IC3 into the internal interrupt code storage register. The code is compared to the interrupt mask bits of the status register. If equal or higher priority than the enabled interrupt level (interrupt code equal or less than status register bits 12 through 15) the TMS9900 interrupt sequence is initiated. If the comparison fails, the processor ignores the request. $\overline{INTREQ}$ should remain active and the processor will continue to sample IC0 through IC3 until the program enables a sufficiently low priority to accept the request interrupt.
IC0 (MSB)	36	IN	Interrupt codes. IC0 is the MSB of the interrupt code, which is sampled when $\overline{INTREQ}$ is active. When IC0 through IC3 are LLLH, the highest external-priority interrupt is being requested and when HHHH, the lowest-priority interrupt is being requested.
IC1	35	IN	
IC2	34	IN	
IC3 (LSB)	33	IN	
			MEMORY CONTROL
$\overline{HOLD}$	64	IN	Hold. When active (low), $\overline{HOLD}$ indicates to the processor that an external controller (e.g., DMA device) desires to utilize the address and data buses to transfer data to or from memory. The TMS9900 enters the hold state following a hold signal when it has completed its present memory cycle. The processor then places the address and data buses in the high-impedance state (along with $\overline{WE}$, $\overline{MEMEN}$, and DBIN) and responds with a hold-acknowledge signal (HOLDA). When $\overline{HOLD}$ is removed, the processor returns to normal operation.
HOLDA	5	OUT	Hold acknowledge. When active (high), HOLDA indicates that the processor is in the hold state and the address and data buses and memory control outputs ($\overline{WE}$, $\overline{MEMEN}$, and DBIN) are in the high impedance state.
READY	62	IN	Ready. When active (high), READY indicates that memory will be ready to read or write during the next clock cycle. When not-ready is indicated during a memory operation, the TMS9900 enters a wait state and suspends internal operation until the memory systems indicate ready.
WAIT	3	OUT	Wait. When active (high), WAIT indicates that the TMS9900 has entered a wait state because of a not ready condition from memory.
			TIMING AND CONTROL
IAQ	7	OUT	Instruction acquisition. IAQ is active (high) during any memory cycle when the TMS9900 is acquiring an instruction. IAQ can be used to detect illegal op codes.
$\overline{LOAD}$	4	IN	Load. When active (low), $\overline{LOAD}$ causes the TMS9900 to execute a nonmaskable interrupt with memory address $FFFC_{16}$ containing the trap vector (WP and PC). The load sequence begins after the instruction being executed is completed. $\overline{LOAD}$ will also terminate an idle state. If $\overline{LOAD}$ is active during the time $\overline{RESET}$ is released, then the $\overline{LOAD}$ trap will occur after the $\overline{RESET}$ function is completed. $\overline{LOAD}$ should remain active for one instruction period. IAQ can be used to determine instruction boundaries. This signal can be used to implement cold-start ROM loaders. Additionally, front-panel routines can be implemented using CRU bits as front-panel-interface signals and software-control routines to control the panel operations.
$\overline{RESET}$	6	IN	Reset. When active (low), $\overline{RESET}$ causes the processor to be reset and inhibits $\overline{WE}$ and CRUCLK. When $\overline{RESET}$ is released, the TMS9900 then initiates a level zero interrupt sequence that acquires WP and PC from locations 0000 and 0002, sets all status register bits to zero, and starts execution. $\overline{RESET}$ will also terminate an idle state. $\overline{RESET}$ must be held active for a minimum of three clock cycles.

STATUS REGISTER	
BIT	NAME
ST0	LOGICAL GREATER THAN
ST1	ARITHMETIC GREATER THAN
ST2	EQUAL
ST3	CARRY
ST4	OVERFLOW
ST5	PARITY
ST6	XOP
ST7 – ST11	not used (=0)
ST12 – ST15	INTERRUPT MASK

Table 2: Status Register Format. The 16 bit status register contains flags which are set during execution of programs. These flags in turn can be tested and manipulated under program control.

mand driven IO interface. Three dedicated IO pins (CRUIN, CROUT, CRUCLK) and 12 bits of the address bus (A3 to A14) are used to interface to the CRU system, allowing input and output bits to be addressed individually or in fields of one to 16 bits. Any bit in the CRU array may be set, reset, tested, or moved between memory and the CRU data field by any of the various CRU single or multiple bit instructions using bits 3 to 14 of workspace register 12 to address the desired CRU bits. When executing multiple bit CRU operations, the data bits are right justified in byte or word locations depending on the number of data bits. The addressing scheme used for multiple bit CRU transfers results in an order reversal of the bits; bit 15 of the memory word (or bit 7 if a byte) is the first bit to be sent or the first bit received while bit 0 becomes the last possible bit in the CRU field. Figure 6 illustrates the bit reversal for multiple bit transfers while figure 7 shows how to construct a typical 16 bit IO port. By decoding the CRU addresses, up to 256 such 16 bit interface registers may be implemented. Typically, only the exact number of interface bits required for a specific device will be utilized.

Several timing and control signals are available for use in implementations of ROM loaders, front panel service routines, a processor hold condition, slow memory cycles and DMA transfers. A wait state is available for use with DMA or slow memory. Table 1 includes detailed descriptions of each timing and control signal that is available.

Five external instructions (CKON, CKOF, RSET, IDLE, LREX) provide a means of initiating any desired user defined external functions. IDLE will also cause the TMS9900 to enter the idle state until an interrupt, $\overline{\text{RESET}}$, or $\overline{\text{LOAD}}$ occurs. When any of these five instructions are executed, a unique three bit code will appear on bits A0 through A2 on the address bus along with a CRUCLK pulse.

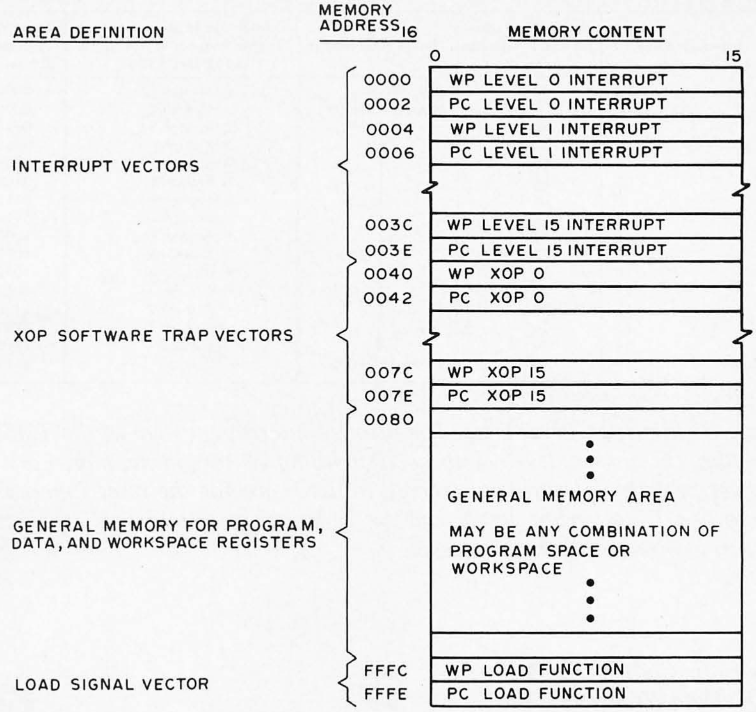


Figure 4: Memory Allocations of the TMS9900. The TMS9900 has a large complement of dedicated locations in memory address space. Interrupt vectors are found in the first 32 words of memory; XOP instruction trap vectors are found in the second contiguous block of 32 words; memory addresses FFFC and FFFE (hexadecimal) are reserved for the load signal vector which defines the location and workspace address of the first program to receive control at system initialization time. The rest of memory address space (hexadecimal locations 0080 to FFFB) is available for general purpose use as RAM or ROM depending upon system design details.

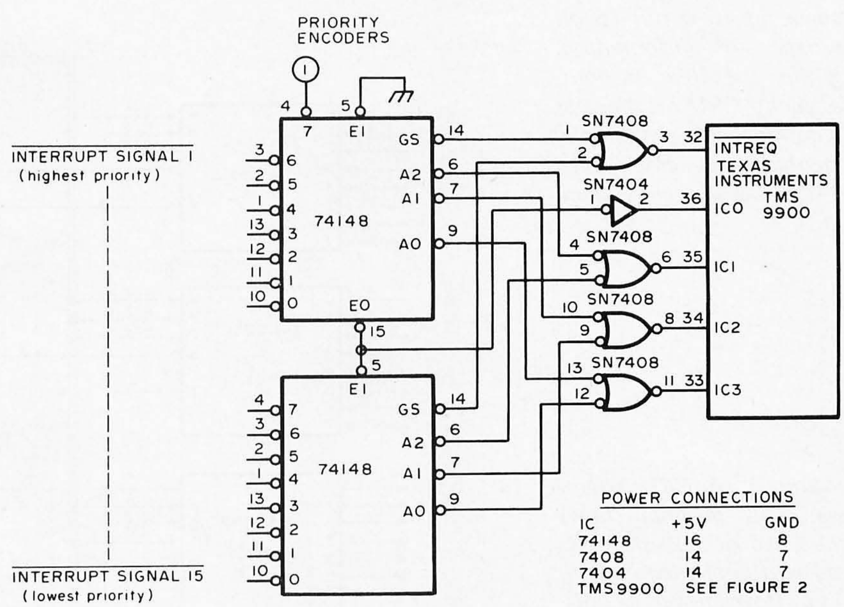


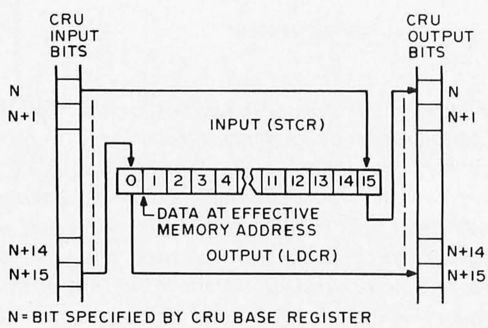
Figure 5: Typical Interrupt Priority Logic. The TMS9900 uses memory address space locations hexadecimal 0000 to 003F as interrupt vectors. One of the 16 double word interrupt vectors is picked by the binary pattern found on lines IC0 to IC3 at the time of the $\overline{\text{INTREQ}}$ signal. This circuit shows one way to generate interrupt vector codes using two priority encoders (74148) and some miscellaneous gates.

Interrupt Level	Vector Location (Memory Address in Hex)	Device Assignment	Interrupt Mask Values to Enable Respective Interrupts (ST12 thru ST15)	Interrupt Codes ICO thru IC3
(Highest priority) 0	00	Reset	0 through F*	0000
1	04	External Device	1 through F	0001
2	08		2 through F	0010
3	0C		3 through F	0011
4	10		4 through F	0100
5	14		5 through F	0101
6	18		6 through F	0110
7	1C		7 through F	0111
8	20		8 through F	1000
9	24		9 through F	1001
10	28		A through F	1010
11	2C		B through F	1011
12	30		C through F	1100
13	34		D through F	1101
14	38		E and F	1110
(Lowest priority) 15	3C	External Device	F only	1111

*Level 0 can not be disabled.

Table 3: Interrupt Level Data. The priority interrupt system of the TMS9900 provides 16 separate levels with corresponding vectors in main memory. The highest priority interrupt is reserved in hardware for the reset function. All the other 15 interrupt levels can be assigned to external devices when a system is wired using this processor.

Figure 6: Control Register Unit Data Transfers. The TMS9900 uses the concept ***of a 4096 bit Control Register Unit address space for programmed IO transfers.*** *This is a bit addressable space accessed by the STCR and LDCR instructions. Transfers to and from the CRU map according to the diagram shown here, and can involve up to a full 16 bit word of information. Within the 4096 bit limit, as many CRU locations as required can be implemented in any given TMS9900 oriented system.*



Instruction Set

The TMS9900 instruction set, as shown in table 4, consists of 67 instructions. These instructions include arithmetic, logical, comparison or manipulation operations on data; loading or storage of internal registers; data transfer between memory and external devices via the CRU; and processor control functions.

The individual instruction execution times are a function of the clock cycle time, the addressing mode used, and the number of wait states required per memory access. There are eight different addressing modes available for addressing both random and formatted memory data as illustrated in figure 8. Not all addressing modes, however, are usable with every instruction.

System Design

A typical minimum system is shown in figure 9 with a single 8 bit IO port. A total package count of 15 packages includes the chips for 1024 words of ROM and 256 words of RAM memory. A general larger, more flexible system is illustrated in figure 10. The clock generator and control section would include the memory decode logic and synchronization logic as well as the clock electronics. Buffers would be required as indicated on all of the system buses to drive the increased loads. Various data and control interfaces are shown in general form since these would usually be designed for specific applications. As with many of the more

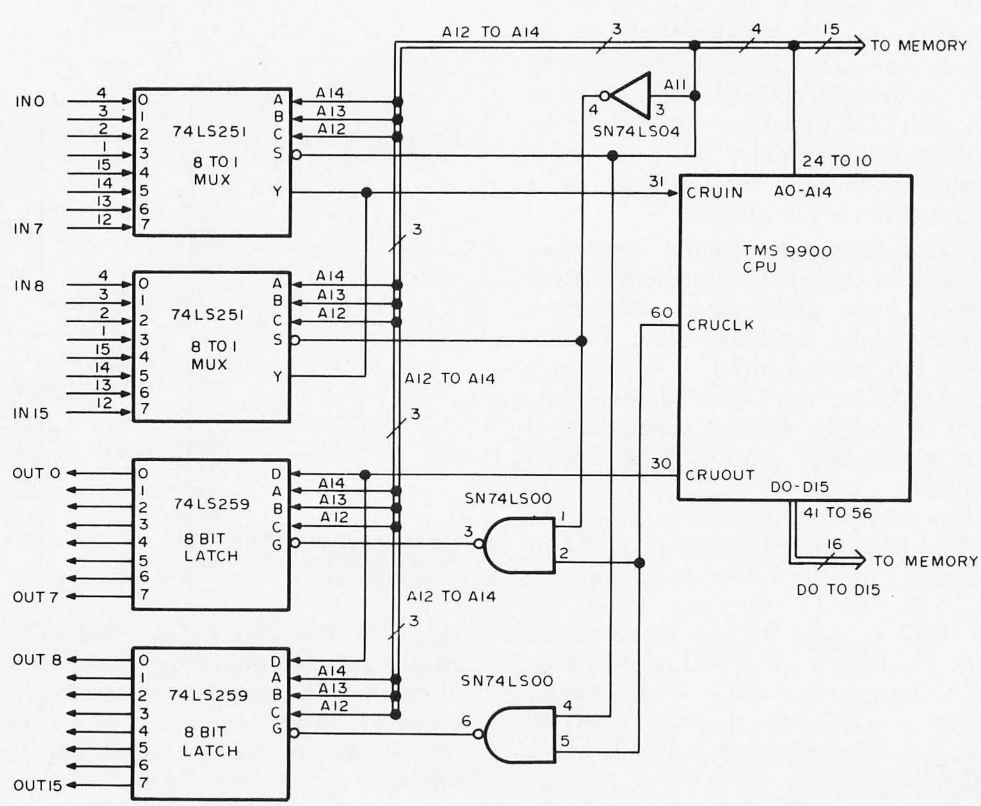


Figure 7: A CRU Example. This diagram shows the logic of a 16 bit CRU segment implemented with TTL (low power Schottky) integrated circuits. Note that a drawing convention of double lines with notation of the number of signals is used to represent multiple path bus interconnections.

advanced microprocessor companies, Texas Instruments also sells the TMS9900 in packaged form as illustrated in photo 2.

Conclusions

The TMS9900 appears to be a powerful 16 bit microprocessor with many features of interest to the computer hobbyist. The advanced interrupt and IO system in addi-

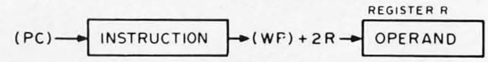
MNEMONIC	DESCRIPTION
AB	Add bytes
ABS	Absolute value
AI	Add immediate
ANDI	AND immediate
B	Branch
BL	Branch and link
BLWP	Branch and load workspace pointer
C	Compare
CB	Compare bytes
CI	Compare immediate
CKOF	User defined
CKON	User defined
CLR	Clear operand
COC	Compare ones corresponding
CZC	Compare zeroes corresponding
DEC	Decrement
DECT	Decrement by two
DIV	Divide
IDLE	Idle processor
INC	Increment
INCT	Increment by 2
INV	Invert
JEQ	Jump equal
JGT	Jump greater than
JH	Jump high
JHE	Jump high or equal
JL	Jump low
JLE	Jump low or equal
JLT	Jump less than
JMP	Jump unconditional
JNC	Jump no carry
JNE	Jump not equal
JNO	Jump no overflow
JOC	Jump on carry
JOP	Jump odd parity
LDCR	Load communication register
LI	Load immediate
LIMI	Load interrupt mask
LREX	User defined
LWPI	Load workspace pointer immediate
MOV	Move
MOVB	Move bytes
MPY	Multiply
NEG	Negate
ORI	OR immediate
RSET	Reset
S	Subtract
SB	Subtract bytes
SBO	Set bit to one
SBZ	Set bit to zero
SETO	Set to ones
SLA	Shift left arithmetic
SOC	Set ones corresponding
SOCB	Set ones corresponding bytes
SRA	Shift right arithmetic
SRC	Shift right circular
SRL	Shift right logical
STCR	Store communication register
STST	Store status register
STWP	Store workspace pointer
SWPB	Swap bytes
SZC	Set zeroes corresponding
SZCB	Set zeroes corresponding bytes
TB	Test bit
X	Execute
XOR	Exclusive OR

Table 4: TMS9900 Instruction Set. This is an alphabetical list of the mnemonics and description of each instruction. The details of operation codes and addressing modes available for each instruction are found in TI literature on the system.

Figure 8: Addressing Modes of the TMS9900. The TMS9900 has a very complete set of memory address calculation modes available to its various instructions. Not all the modes are available to every instruction.

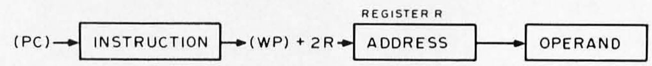
WORKSPACE REGISTER ADDRESSING R

Workspace Register R contains the operand.



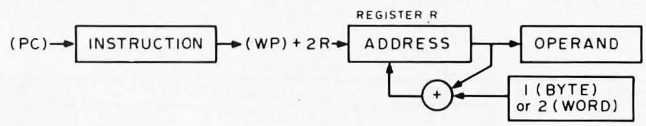
WORKSPACE REGISTER INDIRECT ADDRESSING *R

Workspace Register R contains the address of the operand.



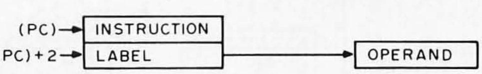
WORKSPACE REGISTER INDIRECT AUTO INCREMENT ADDRESSING *R+

Workspace Register R contains the address of the operand. Upon completion of the operation, the contents of workspace register R are incremented.



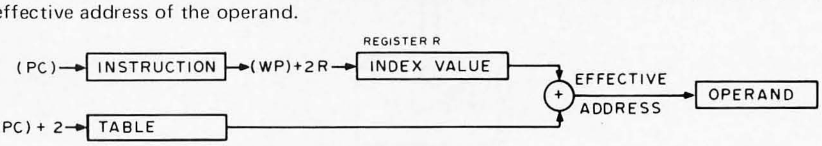
SYMBOLIC (DIRECT) ADDRESSING @ LABEL

The word following the instruction contains the address of the operand.



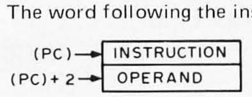
INDEXED ADDRESSING @ TABLE (R)

The word following the instruction contains the base address. Workspace register R contains the index value. The sum of the base address and the index value results in the effective address of the operand.



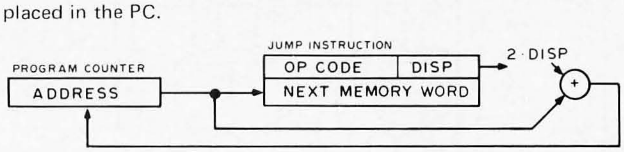
IMMEDIATE ADDRESSING

The word following the instruction contains the operand.



PROGRAM COUNTER RELATIVE ADDRESSING

The 8 bit signed displacement in the right byte (bits 8 through 15) of the instruction is multiplied by 2 and added to the updated contents of the program counter. The result is placed in the PC.



CRU RELATIVE ADDRESSING

The 8 bit signed displacement in the right byte of the instruction is added to the CRU base address (bits 3 through 14 of the workspace register 12). The result is the CRU address of the selected CRU bit.

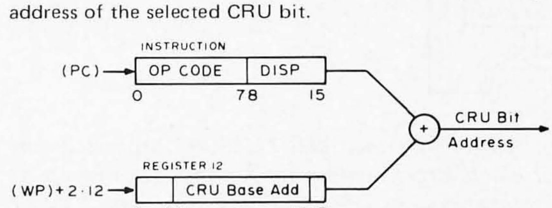
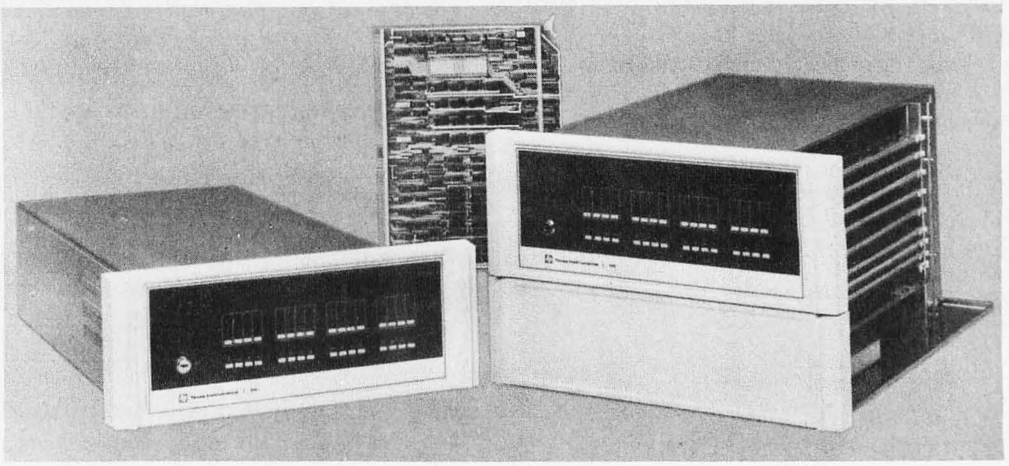


Photo 2: As is the case with a number of the microprocessor manufacturers, Texas Instruments supplies the TMS9900 part separately or integrated with other components to form the card and cabinet oriented modules illustrated here. For industrial and commercial users who desire even higher speed computing, there is also a more expensive (but software compatible) version implemented with ordinary TTL logic.



tion to flexible memory and processor control make many desired system designs simple to implement. Also, the workspace concept may provide several new software capabilities that are greatly needed in the microcomputer field today. As a final comment in passing: IBM 360/370 lovers should take note of the similarities between the TMS9900 general register architecture and the IBM large scale computers.■

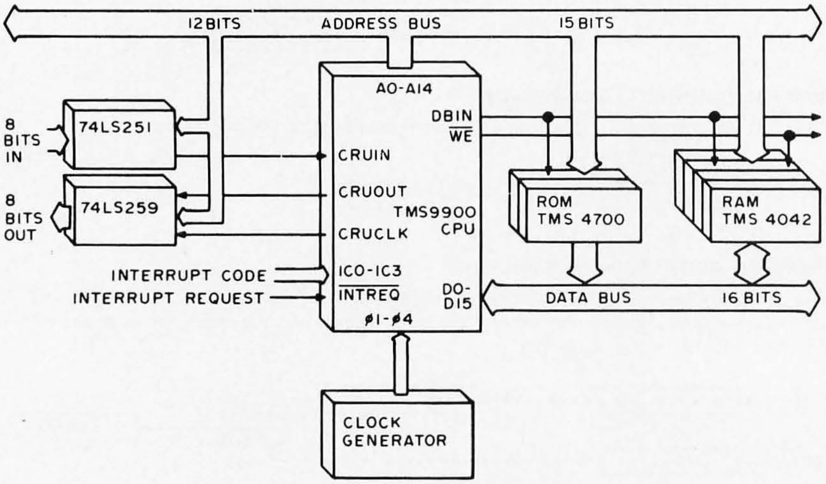


Figure 9: A Small System. The TMS9900 design, while quite general purpose in nature, does not necessarily have to be built into a general purpose system. A small system might consist of (for example) 8 bits of CRU, the CPU, clock generation modules, a small amount of RAM and some ROM for fixed programs.

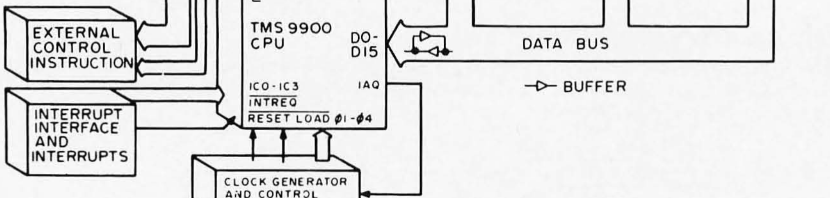


Figure 10: Larger Systems. With buffering, the TMS9900 processor can become the central figure of much larger computing systems, with much of the power of the traditional minicomputer. Here is a sketch of such a larger system.

GLOSSARY

Contiguous: Adjacent in sequential order. Several memory words located at sequential addresses in memory address space are said to be contiguous with one another.

Context changes: When it is necessary to begin a new program upon interrupting or returning from an executing program, the state of the CPU changes. This is a context change. The TMS9900 provides a very efficient and complete context change mechanism in hardware which guarantees that all working registers and status information are changed in an orderly fashion.

Extended Operations (XOP): The TMS9900 contains a facility not found in many of the early microcomputer designs: a set of XOP instructions which allow 16 user-programmed functions to be called; the XOP's effective address calculation is used to set up parameters which are available to the XOP service routine. A special set of trap vectors is accessed by the XOP instructions.

Memory to Memory: Many microprocessors to date have employed an accumulator oriented architecture which requires at least one operand to be an on chip register; in contrast, the memory to memory design of the TMS9900 is oriented toward the use of two memory operands.

Process swapping: See context changes.

Trap: A technique of computer design which provides a mechanism to detect certain software conditions and cause interrupts. A trap vector is a set of information stored in the computer's memory for purposes of accessing a subroutine designed to handle the trap condition. For the TMS9900 trap a vector consists of the address of the subroutine and the address of the workspace for the subroutine.

continued from page 10

quite useful. I enjoyed the series on LIFE in particular. Being a LIFE addict from way back, I shall have to implement the game on my 6800 system once it comes online.

There is at least one thing that I have noticed missing from among the articles, and that is something on the use of a micro for the purpose of sending and receiving RTTY on the amateur bands. Being an amateur myself, and involved in VHF FM work, I plan on setting up the necessary hardware and software to use my system as a radio Teletype converter.

Along the same lines, I also plan on writing the programs necessary to transliterate hand sent Morse code to printed letters. The little information that I have about the algorithms to do this with are gleaned from a *Scientific American* article circa 1960, so perhaps there have been better ways discovered since then. If so, I wonder if any of your readers might clue me in to some references on the subject of computer pattern recognition, which is the basis for any work such as this.

Keep up the good work, and I will be watching with interest to see where things go from here.

Richard Fall
PO Box 3486
Stanford CA 94305

COMMENTS ON COMPUTER MEGALOMANIA

A Von Mises fan, I learned my lesson the easy way ("A Lesson in Economics"), having started my subscription with BYTE #1. Congratulations on the excellent job you have done to date.

Suggestion: put BOMB & the BYTE questionnaire on a card — I'm not about to rip up my BYTE or travel two miles to the nearest Xerox machine to make one copy. I know of all the languages on your questionnaire and have used most of them, but I noticed one glaring omission: APL. It is one of the best languages in existence (in terms of conciseness, consistency, flexibility, ease of use and scope of application).

The author of "Could a Computer Take Over?" made the comment: "Computer technology is already far outstripping man's capability of harnessing it." No way! There are many problems (and more turning up daily) from the fields of mathematical physics or molecular biochemistry (just to name a few) that can easily swamp computers a trillion times faster than today's fastest computers. Another comment made was: "Would government by computer really be all that bad? In a case such as that in *The*

Moon Is A Harsh Mistress, the answer would be no." However, this is not a case of computer as government but computer as guerilla since MIKE (the computer) helped Professor De La Paz (the rational anarchist) overthrow the existing government and prevent any "real" government from reemerging. Nevertheless I liked the article.

Keep up the good work. Best regards from Silicon Valley,

Conrad Schneiker
Mountain View CA

Readers will note that reader's service and subscription forms are incorporated as a separate card beginning this month. The economics of mass production prevent putting BOMB or the questionnaire on the card.

WHO ARE YOU TRYING TO KID?

I got my February issue of BYTE recently, my first issue. You have an excellent magazine, well worth the subscription price.

But who are you trying to kid? On page 14 of February BYTE you show a picture of a two story building in which, according to the text, you are located on the fourth floor! What's up?

A Q Smith
St Louis MO



The picture is real, and the text is correct. The entrance to the building shown in February BYTE reaches the third floor, since the building is nestled into a hillside. As evidence of the fourth floor location of our offices, Ed Crabtree recently lay down in a snowbank, pointed his camera toward the sky, and took this picture of our offices from a different vantage point.

JOIN THE CLUB?

I read a very interesting editorial by Carl Helmers in the February issue of BYTE magazine. Its headline is similar to this one. As Mr Helmers described a national personal computer society I started reaching for my checkbook and a self addressed stamped envelope. I don't care what it costs, I want to join. But in the last chapter, as in *Don Quixote*, the Knight of the Mirrors brings reality home with a vengeance. The enemy is us.

My interpretation of Mr Helmers' closing remarks is: If it looks like a duck, walks like a duck, quacks like a duck, it is most certainly a duck even if it calls itself the Southern California Computer Society. Sadly I gave up the vision of the Great Galactic Computer Cavalry riding to our rescue and went to the 942nd daily meeting of the Trivia and Tedium Standards Committee. The world has again proven that if you glue enough hangnails together, you get Frankenstein.

If you glue enough hangnails together, you get Frankenstein.

The stark naked truth is that we have no plausible defense against Mr Helmers' charges. In fact we are lucky he didn't expand the indictment to include "International Society," which we can not deny either. I think it is unfair that he did not include a few mitigating facts, such as our 370-158 group purchase fiasco and the total lack of attendance at our Lunar Chapter meetings. With just a few more well planned, superbly executed disasters, we might have escaped this one.

The dastardly attack on The Society

name seems strange and somehow provincial from an organization aspiring to nationwide scope. At the venerable age of four months the acronym SCCS is at least as recognizable as SPQR, which was a fairly active society in its day. If our name is objectionable: so be it. We do not aspire to the kingdom, the power, and the glory. We just hope to help the next guy who wants to know which is pin 1.

Our first and foremost goal in our charter is to provide the best possible service to anyone who asks. But we demand a heavy fee: the same caliber of service to the next one who asks. On this premise and this name our organization stands, and all are welcome to join. There is no residence requirement nor any other impediment. We are not the new Genghis Khan astride a 40 legged horse called 8080 sweeping out of the West to cut the Gordian Knots of Computerdom. We are building the tools to pry open Pandora's Box and show computers for what they are and what they might become.

The name of our publication, *Interface*, is probably also misunderstood. The computer interface is usually understood to be the common boundary between two subsystems. If the computer interface is proper, the boundary disappears; and the whole becomes greater than the sum of the parts. This is why our *Interface* has become our flag ship. We have a message, we have a voice, we will speak, and we will be heard.

Lest anyone mistake our clarity of purpose for ambition, note that we have not asked Jeff Landry to change his name to Jeff Lendry to preserve our image. Also note that the illustrious founder of *Interface*, Jon Walden, must look down from the Pantheon at the January issue where his name is wrong three times in three tries. The bell tolls for you, Art Childs.

Returning to this scurrilous attack from BYTE; beware, Carl Helmers, the august president of SCCS is not powerless. One more outburst of this nature and we shall resort to the violence vested in me and name you honorary member and chairman of The Colossal Name Committee. (And publish your secret, unlisted phone number. We haven't been infiltrated by the CIA for nothing!) On guard, Sir, defend yourself!

If we can get back to our real business, we have a lot of work to do. We are building one brick at a time and we are competent builders. We will continue to do our best.

Ward Spaniol
President, SCCS
Los Angeles CA

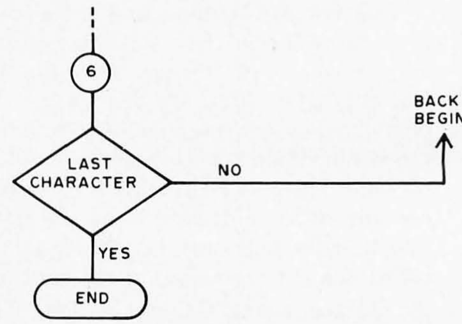
BYTE'S BUGS

Here lies documentation of known bugs detected in previous editions of BYTE . . .

The following correction should be made to the specifications of the HP-10525T found in table 1, page 24, December 1975 BYTE, in Alex Burr's article "Logic Probes – Hardware Bug Chasers." The scale factor on the minimum pulse width specification was misprinted in the table and in note number 6 applying to the width specification. The HP-10525T will respond to pulses as short as 10 ns (nanoseconds) guaranteed, typically as short as 5 ns in favorable circuit conditions. Pulses shorter than .05 seconds are stretched to .05 s. The printed version showed "ms" (milliseconds) which is only 6 orders of magnitude larger than the actual specification printed by HP.

"My Dear Aunt Sally," page 20, February BYTE: Two errors in our preparation of the flow diagram of figure 1 were detected by reader Tim Walsh, and confirmed by author Robert Grappel.

Segment A (Main Routine): A "last character" test was omitted between node 6 (the return point from routines in segments B through F) and the END box. The proper flow for the end of the main routine is:



Segment B (Single Character Function Name Handler): The YES and NO labels on the conditional test "1 CHARACTER OP" were reversed.

Classified Ads Available for Individuals and Clubs

Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed or printed notice to that effect. The notices are free of charge and will be printed one time only on a space available basis. Insertions should be limited to no more than 100 words. Notices can be accepted from individuals or bona fide computer users' clubs only. Commercial advertisers should contact Virginia Peschke at BYTE for the latest rate card and terms.

PDP-11 wanted: Will trade 1000 copies of **Computer Lib/Dream Machines** for one standard PDP-11 minicomputer. Contact Ted Nelson at (312) 352-8796.

FOR SALE: Tape drive, Wangco model 7, 12.5ips, 7 track (change head only for 9 track) unused $500 each. Cassette drive, Computer Access Systems (MFE) model 250, 10-40ips (Altair compatible interface available from Cybertronics) new $250 each. Line printer, Tally model T-132, 100 1pm, 132 column, 300, 600 or 1200 baud serial, well used $500 each. Core memory system, Ferroxcube (FX Systems) model FI-3C-4K12-22, 4Kx12 bits, 1 sec, new $400 each. Write John L Marshall, Box 242, Renton WA 98055.

Tired of waiting for MITS memory boards? I have two MITS 4 K Dynamic Memory Boards for sale at the kit price of $195 each. These boards were assembled by an electrical engineer and factory checked. All bits certified. H S Corbin, 11704 Ibsen Dr, Rockville MD 20852, (301)881-7571.

I will trade up to 1000 hours of programming and systems analysis for a complete system consisting of the following: An 8080-A based CPU with 64 K memory, dual drive floppy disk system, Qume

55 CPS printer, 9600 baud CRT with keyboard (upper and lower case ASCII and at least 12 lines of 80 characters, though 24 lines by 80 would be better). The system should be fully assembled, integrated, tested and with enough basic software (editor, assembler, IO drivers) to make it useful. I have been programming for over 14 years on systems large and small. Charles B Shipman Jr, 220 Sandburg St, Dunn Loring VA 22027.

SWTPC CT-1024, TV Typewriter Main and Memory Boards completely wired. Recently tested by SWTPC. With instruction manual. $150 from Don Gould (313)566-6380 days.

WANTED: Construction manual and data package for the Mark-8 Mini Computer. Appeared in *Radio Electronics* in August 1974. Reply to Robert L Gerald, PO Box 406, NY NY 10013.

TVT for sale, works perfect! Displays 25 lines of 40 chars each. With power supply and cabinet. Asking $175. Also have working boards (to attach to above unit) that accept serial ASCII and serial BAUDOT codes. Asking $35 each. High quality parallel ASCII keyboard, assembled and working, $40. R-E 4 Channel digital memory scope attachment – assembled but not tested, $150. You pay postage, prices negotiable, am willing to trade for some sort of ASCII printer. Jeff Roloff, 2214 Brookshire Dr, Champaign IL 61820, (217)356-0780.

FRIDEN FLEXOWRITER Model SPS. Consists of keyboard, printer, 8 level paper tape reader and punch. Good working condition with schematic. Contact Phil Hughes, PO Box 43, Richland WA 99352. Phone: (509)942-7045 days, (509)946-7938 evenings. $500.

CAELUS Model 303 Disk. I need to locate maintenance and repair manuals. Kurt T Rudahl, Box 6652, San Francisco CA 94101. (415)864-2263

FOR SALE – ICP digital computer cassette decks. These OEM decks are complete and need only power supply and case and use TTL interface logic. May need some slight repair or adjustment. They are International Computer Products DigiDeck model 63 read/write decks with encoder board and 2 track operation. $5 to ICP, Box 34484, Dallas TX 75234, can get a complete technical and maintenance manual. While my supply lasts I will sell these decks for $100 each. Send bank check or money order to Gary Fishkin, Deck, Box 349, 25 Andrews Memorial Dr, Rochester NY 14623.

FOR SALE: Intel SIM8-01 Microprocessor System includes 8008 CPU, TTY interface, IO connectors. Also 10 Intel 1702 PROMS, all never used; original cost $1500; first certified check or MO for $250 takes all. M Siegel, 41 Middle Loop Rd, Staten Island NY 10308.

Friden high speed PAPER TAPE PUNCH, new, in carton, no dust cover, $25! CARD READER, excellent condition, approximately 150 CPM, small simple unit, $35! Please include postage for either unit. Frank Smith, Box 386, Coburg OR 97401.

Monitor, Text Editor, and Assembler for an 8080 available. Total memory requirement is 4 K bytes. I can also burn Intel 2708/8708 and 2704/8704 PROMs for your system. Contact Albert Badu, 105 Fairmont St, Arlington MA 02174, (617)646-4539 (evenings).



BOOK REVIEW

An Introduction to Microcomputers, *Adam Osborne & Associates, Inc, 2950 7th St, Berkeley CA 94710, (415) 548-2805, $7.50.*

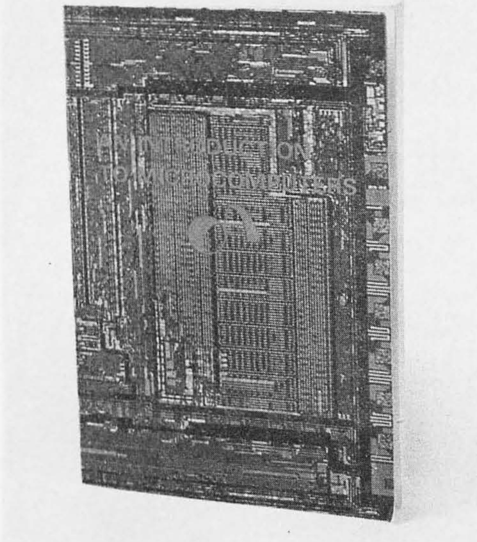
This is an excellent book, useful for both neophytes and experienced electronics designers or programmers (I am the latter). It focuses on instruction sets possible and available, as well as external logic system considerations. About a third of this 406 page book is devoted to detailed, objective examination of six microcomputers: the Fairchild F8, National PACE and SC/MP, Intel 8080, Motorola 6800, Rockwell PPS-8, and Signetics 2650.

The first two chapters (16 pages) contain some computer history and background on binary arithmetic and logical operations. This is standard material, such as two's complement arithmetic, De Morgan's theorem, etc. The next chapter (22 pages) explains memory organization and addressing, with differences between ROM and RAM usage mentioned. It also discusses how memory data is interpreted: as a binary or BCD number for single and multibyte operations, as a character code, or as an instruction code.

Having covered the fundamentals, the next chapter (65 pages) is devoted to the generalized microcomputer itself – its architecture, control, and timing. The various internal elements are described, such as accumulators, program counter, other specialized registers, the arithmetic/logic unit, and status flags. Throughout the book, many examples are used to clarify explanations; for example, two pages show the various ways overflow status is usually generated. Next the basic fetch/execute CPU cycle is shown, with sample timing diagrams related to an illustrative chip pinout. This section is treating that mysterious boundary between hardware and software. Exactly how instructions are executed is treated in detail with 15 pages on the microprogramming of register transfer sequences.

The last two generalized chapters delve further into system hardware (49 pp) and programming (106 pp). The hardware chapter starts with more on RAM and ROM interfacing, using some hypothetical bus systems. Most of this section deals with input/output operations. Interrupt processing and direct memory access operations are discussed extensively. Frequent use is made of illustrative timing diagrams and chip interconnections; however, actual IO devices (displays, cassettes, etc.) are never mentioned. A brief discussion of serial IO concludes the chapter.

The programming chapter creates a hypothetical instruction set, explaining the rationale for various addressing modes and instruction operations. It starts with an introduction to programming and assembly language. One word of warning here: The authors view microcomputers more as replacing hard logic than as super cheap general purpose computers. They view "minicomputer" and "microcomputer" instruction sets in this light. However, hobbyists frequently are interested in running many different programs, and in ease of programming, which aren't important in systems implementing logic for industrial controls or washing machines. For example, many instruction sets assume the program resides in ROM, although hobbyist programs are more often in RAM. There are other, more subtle distinctions. Subroutine argument passing, and the ability to address memory based on full 16 bit pointers in registers or indirectly in RAM, are very important in general purpose systems. Still, this chapter is valuable when specific microcomputers are discussed. Common termi-



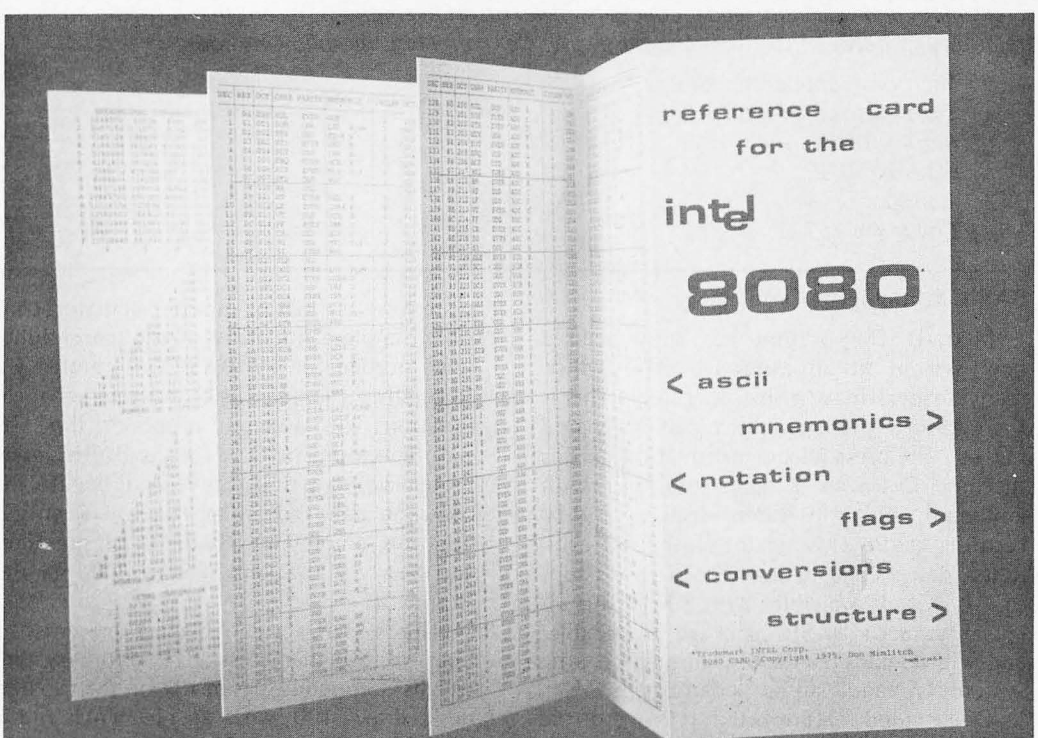
nology is established, and a coherent instruction set organization makes comparisons much easier. I wish their sample instruction set was available in some microcomputer!

Finally the reader arrives at the heart of the book – descriptions of the six microcomputers. Although so far only the PACE, 8080 and 6800 have appeared as hobbyist kits, the others are worth looking at. The F8 and 2650 both use very simple clocks and have an interesting organization. The PPS-8 is probably least suitable as a hobbyist microcomputer, due to a strange and overly complicated instruction set. Some hobbyist chips are not covered; notably the MOS Technology 6502 and Intel 8008; but also such new chips as the 16 bit General Instrument CP1600 and Texas Instruments TMS9900, or the CMOS Intersil IM6100 and RCA CDP1801.

Each microcomputer is outlined in the same way: features, registers, and flags; CPU pins and bus signals, accessory chips available, interrupt and direct memory access methods, and a complete chart of instructions in standard format. There is enough data to pick one best microcomputer for a given application, but manufacturer's literature would be needed to actually build a system. A common benchmark program is implemented with each chip, although the authors wisely caution against using these as representative of efficiency, which varies greatly with the application and the assumed conditions. These sections give some feel for the programming philosophy of the chip designers, however. For choosing a microcomputer for your system, this book is an excellent place to start. ■

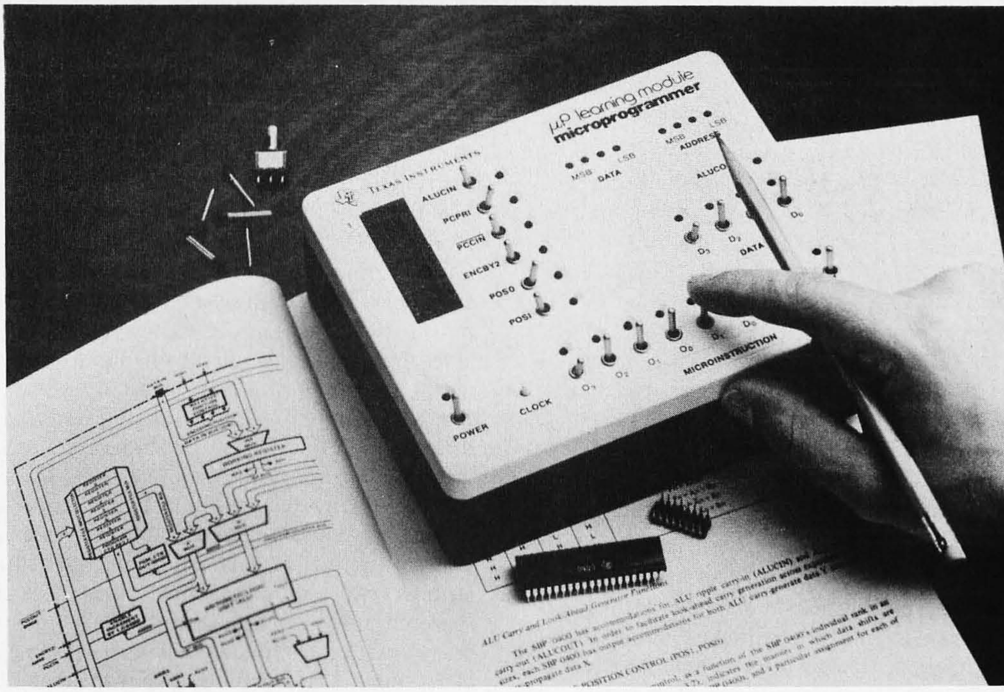
Bob Wallace
PO Box 5415
Seattle WA 98105

BYTE'S BITS



Need an 8080 Reference Card? This is the 8080 reference card which has been prepared by reader Don Mimlitch, 29 Hoyt St, Stamford CT 06905. It is available from Don for $1.25 each, or 5 for $5 as a special introductory offer. The sample card which Ed Crabtree photographed for us has proven quite useful in preparing articles with 8080 listings for BYTE, and it will no doubt become a valued addition to any 8080 user's documentation tools.

What's New?



TI Microprocessor Learning Module: Here's a new item of considerable interest to readers desiring to learn the fundamentals of programming and microprogramming with a self contained battery operated processor which uses calculator packaging technology. The module and documentation come as a completely assembled ready to use package.

Quoting the Texas Instruments press release, "A hobbyist, engineer or technician using this module about two hours a day could derive more than six weeks of concentrated self education." The price is $149.95 in single quantities. It was developed by a team of TI microprocessor experts and members of the academic community to teach basic microprogramming and machine code concepts. The tutorial emphasis is on the difficult to grasp relationship between software and hardware. Contact TI at the following address:

Microprogrammer Modules
Texas Instruments Inc
Inquiry Answering Service
PO Box 5012
Mail Station 84
Dallas TX 75222

BASIC-8 Intelligent Terminal

Mikra-D Corporation has announced a new system at an attractive price which many readers may wish to consider. This is the BASIC-8 Intelligent Terminal, designated MTS-8. The press release information on this machine describes a table top unit with keyboard, built in video display, a dual recorder audio cassette interface, and cabinetry, delivered with software. Two versions of BASIC are available. With 8 K of memory, a standard BASIC package is available; and with expansion to 12 K of memory, the system can handle an Extended BASIC.

One unique feature is that it is one of the first "multimicroprocessor" systems available in package form for the user. Its main processor is an 8080 device. This is the processor which runs the ROM operating system, the BASIC interpreter, and applications programs. The second processor of the system is a floating point calculator chip which implements many of the special functions needed for the BASIC interpreter. By calculating the scientific and mathematical functions with a second processor, the software development time for the BASIC interpreter was cut drastically, and the BASIC user can take advantage of full scientific notation employing 10 digits of precision and nearly 200 orders of magnitude for the exponent.

The system's video display presents 24 lines of 80 characters. The entire system weighs 35 pounds, sits neatly on the top of a desk, draws 100 watts at 110 VAC, and is available in kit form for $1695 in the 8 K BASIC version. The 12 K Extended BASIC version of the kit is available for a total price of $2230. Assembled and tested versions are also available. Contact Mikra-D at 770 Washington St, Holliston MA 01746.

IO Strobes for the Altair 8800

John M Schulein
1186 Arlington Ln
San Jose CA 95129

Figure 1 shows a simple way to generate eight IO strobes for an 8080 based microcomputer system. It uses only four integrated circuits and is shown with the MITS Altair signal names and connector pins for the input signals. When the SOUT signal is active high, the 7442 is steered to the lower output strobe codes (0, 1, 2 and 3) and the $\overline{\text{PWR}}$ signal generates one of the output strobes when it hits pin 12 of the 7442. For inputs, the SINP signal is active high and SOUT is low, so that the 7442 is steered to the higher input strobe codes (4, 5, 6 and 7). When the PDBIN signal hits pin 12 of the 7442, an input strobe is generated. The 8131 integrated circuit is a comparator that is configured with six jumpers to select a group of four IO ports. Note that the selected four IO ports are contiguous and start at 0, 4, 8, C, 10, etc. (Hex), depending upon the binary coding of the six 8131 jumpers.■

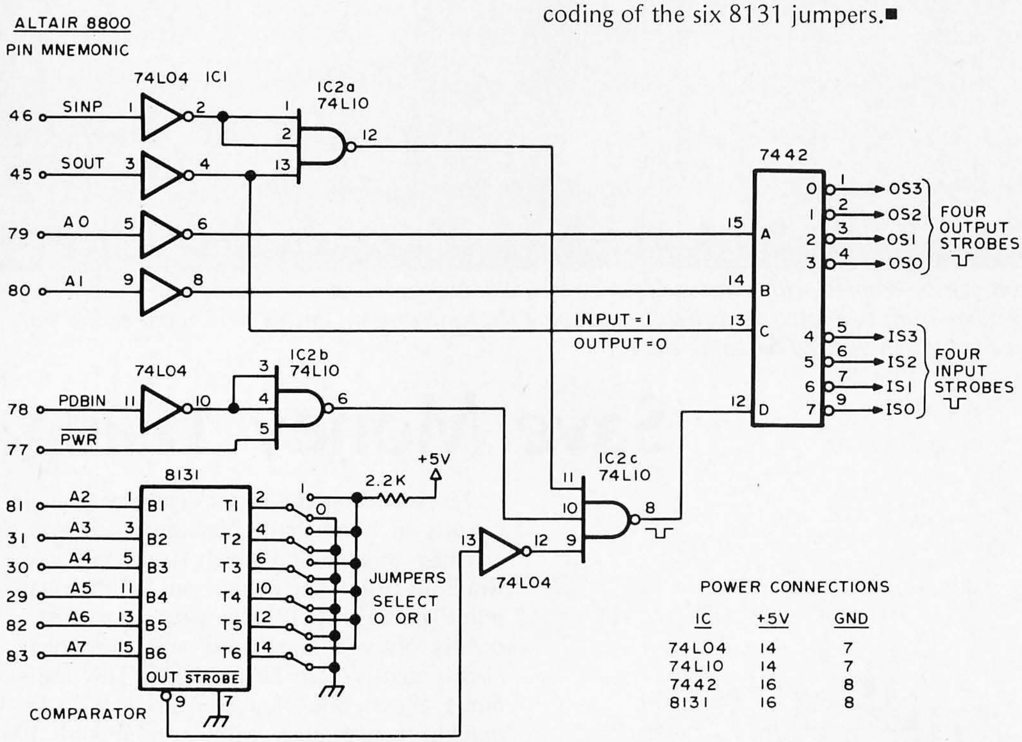


Figure 1: How to generate four Altair 8800 output strobes and four input strobes using four integrated circuits.

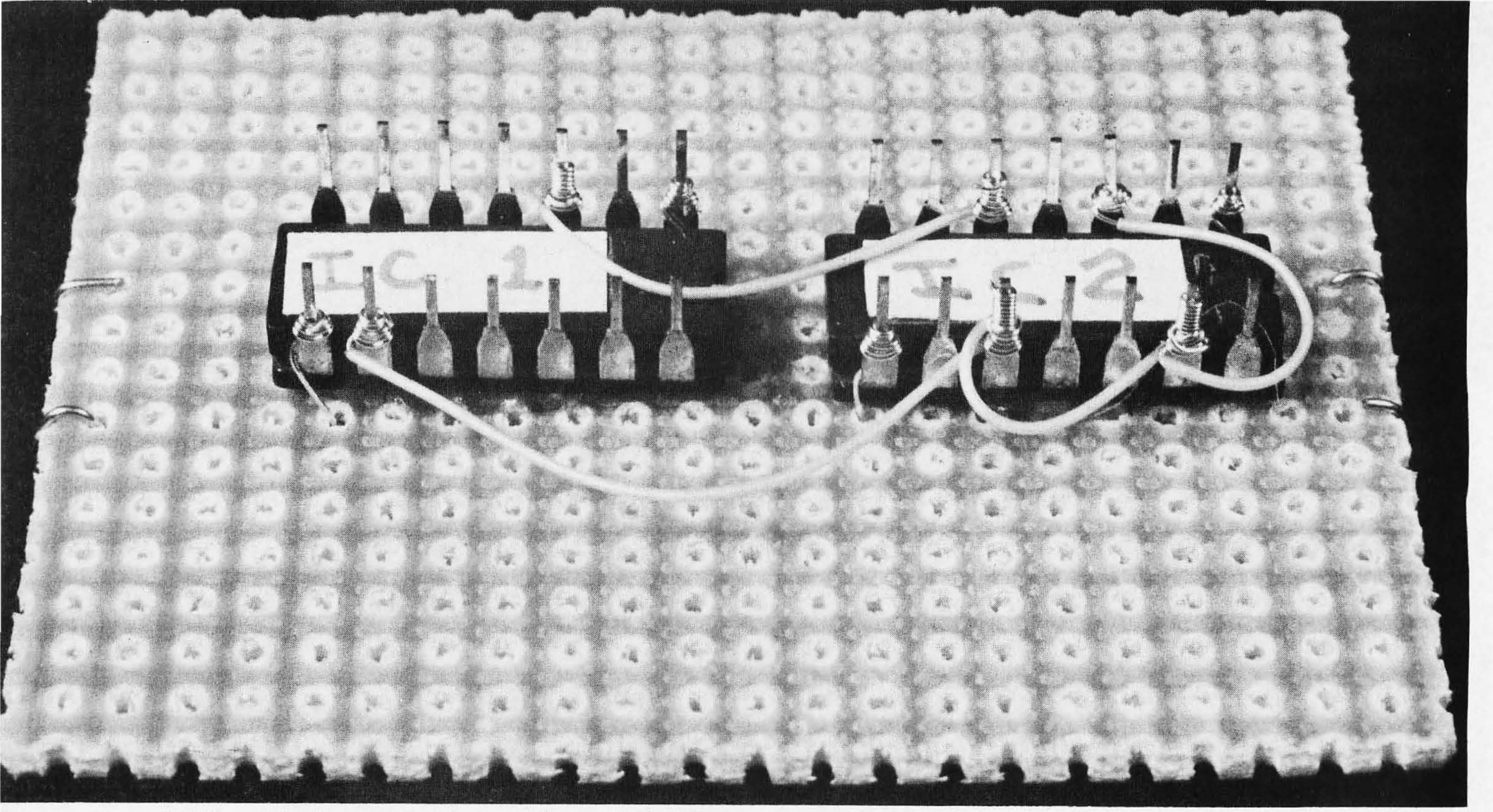
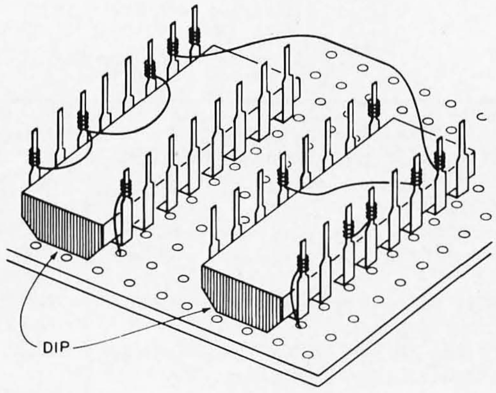


Photo 1: The mini wire wrap method can be applied with some success by virtually any experimenter without large capital investment. Here is a sample assembly made from two integrated circuits and some Vector P pattern unclad perforated board. For extra rigidity of the mounting, a spot of glue can be used to secure the IC.

Save Money Using

Roger W Thompson
5950 Valkeith
Houston TX 77035



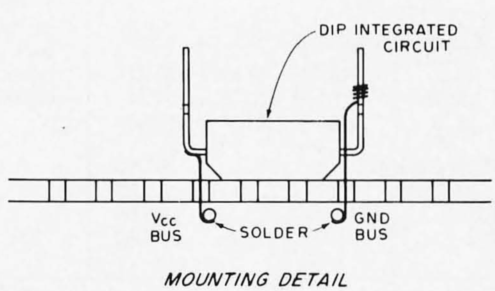


Figure 1: Mini Wire Wrap Construction. The integrated circuits are mounted without sockets, upside down on a perforated board base. The mounting detail shows how the IC is attached by using a short jumper through the board.

For real economy in packaging integrated circuits in home built breadboard designs, consider this method which I call mini wire wrap. By using the actual pins of the dual inline package (DIP) as wire wrap posts, connectors are eliminated and very high circuit density can be obtained. This technique allows one of a kind projects to be quickly constructed with a minimum of material. The mini wire wrap method uses perforated phenolic or epoxy glass board. Integrated circuits can be mounted with the pins sticking up using mechanical fastening of the Vcc and ground pins soldered to bus wires on the opposite side of the board. (See figure 1.) A small amount of glue can also be used if a stronger mounting is desired.

Tool Construction

A tool is required to wrap the wire around the IC pins. This can be constructed from a metal ball point pen cartridge. Cut the ball from the cartridge and flush the remaining ink with a solvent. As shown in figure 2, a small hole (0.030 inch, 0.762 mm) is drilled through one side of the tube approximately 0.030 inch, 0.762 mm from the end. The end of the tube and the side hole must be deburred. The larger end of the tube (0.125 inch, 3.17 mm) can be chucked

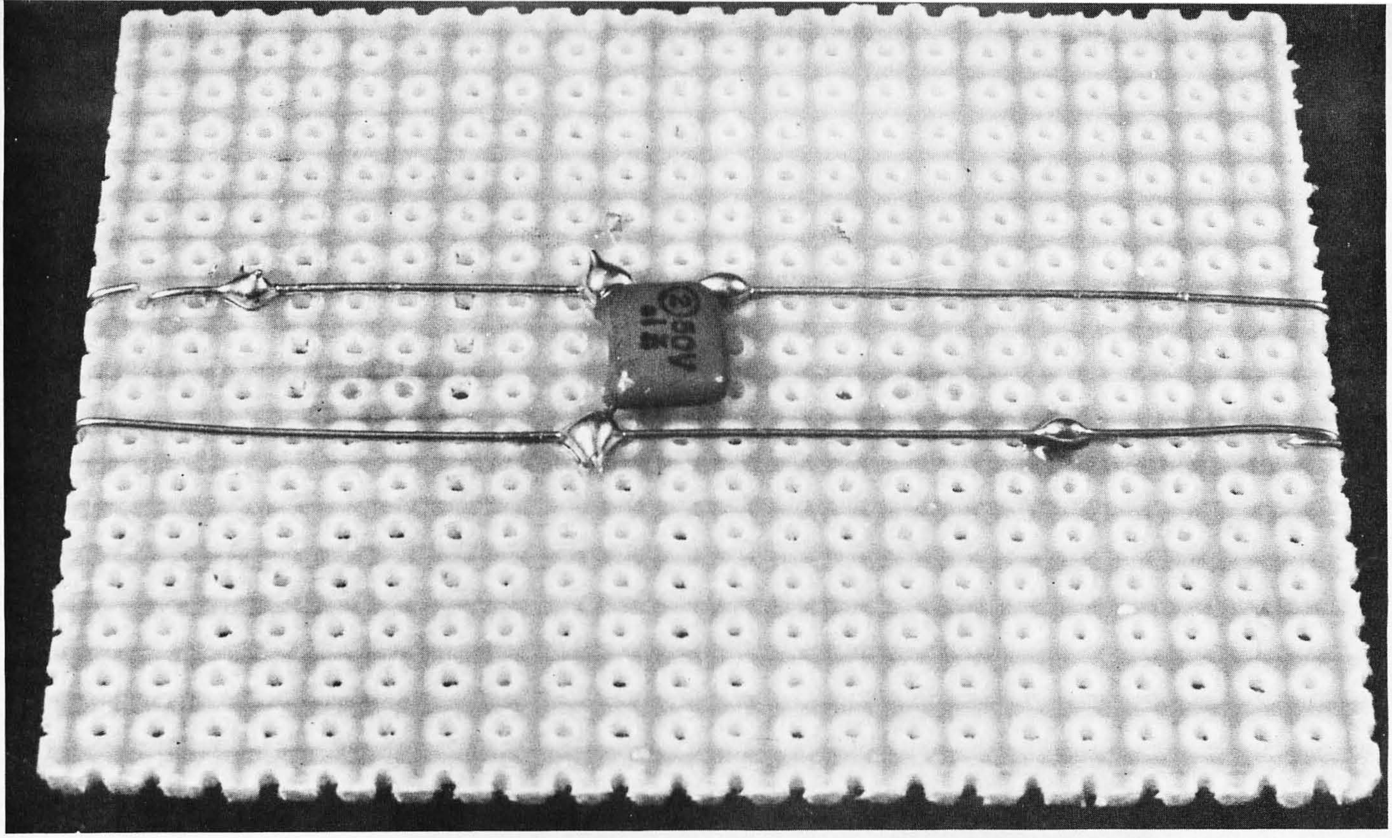


Photo 2: The reverse side of the assembly in photo 1 shows how two bus strips have been mounted. These strips have connections through the board to the power and ground terminals of the ICs. Also shown in this sample assembly is the bypass capacitor which is part of normal power distribution practices for logic families.

in a tool handle such as those provided for small multipurpose screwdrivers. An unwrap tool can be fashioned from a small piece of hard vinyl tubing, or the tip of a thin lead mechanical pencil, with an inside diameter of approximately 0.030 inch, 0.762 mm. A notch cut in the end provides all that is necessary to loosen the wire on a previously wrapped pin.

The hookup wire used should be ordinary computer wire wrap wire: 30 gauge solid wire with thin wall insulation.

A wire stripper of the scissors type with an adjustment screw will provide satisfactory stripping operation. Adjust it so that few nicks occur.

Operation

To use, strip about 0.375 inch, 9.52 mm of insulation from a piece of wire and insert the wire in the wire wrap tool with the bare end sticking out of the side hole. Slip the tool, with wire inserted, over a DIP pin and twist the tool to wrap two or three turns around the pin. Care should be taken not to apply excess pressure since the IC pin can twist and break. Two wires can be connected to one pin to allow chained interconnections. Leave a little slack in wires between integrated circuits so that replacement will be quicker, if it should become necessary. Generally speaking, integrated circuits are quite reliable. Unless an IC is misapplied (e.g., Vcc and Gnd reversed), replacement should be rare in small circuits (10 to 30). Since the DIPs are mounted upside down, it may be helpful to mark each DIP with its identity since the manufacturer normally marks the other side.

I have used this technique successfully with TTL and CMOS digital ICs as well as linear types. No intermittents have been encountered in several years of use. Circuit density of four DIPs per cubic inch can be achieved, which is comparable to multi-layer printed circuit board packaging. After a circuit has been debugged, or if a bad connection is suspected, the wires can be optionally soldered to the pins with a very low wattage fine tip soldering iron. The mini wire wrap technique may not measure up to the rigorous standards of regular wire wrap, but it works and it may cut the cost of a project in half. ■

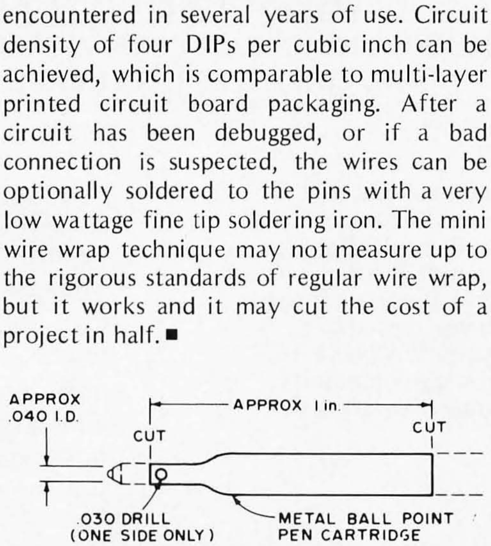
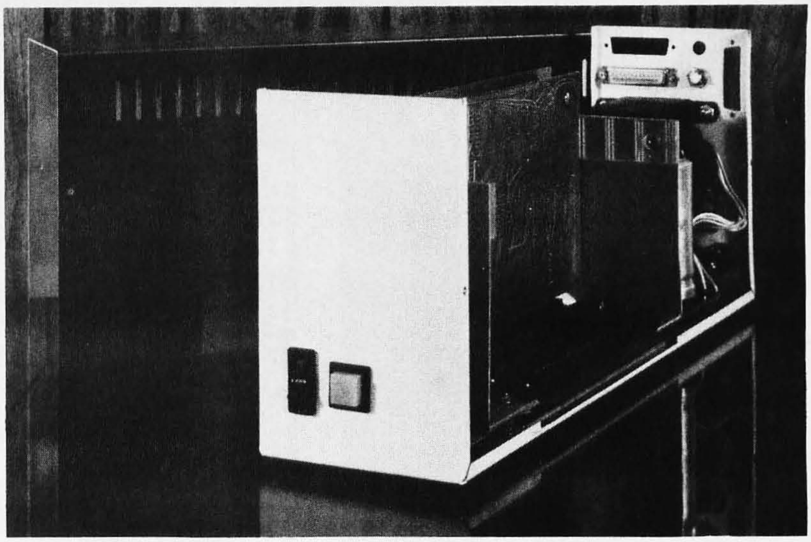


Figure 2: Mini Wire Wrap Tool. One way to make a tool useful in this technique is to recycle a metal ball point pen cartridge. Remove the point, flush out the residual ink with a solvent, then drill a small wire guide hole in the narrow portion of the cartridge. After deburring the holes, the tool can be mounted in a home made handle or the holder of an interchangeable blade screwdriver set.

BYTE'S BITS



NEWS FROM THE TRADE: MITS, POLYMORPHIC

MITS has outgrown its facilities again and has moved to larger quarters. Their new address is 2450 Alamo SE, Albuquerque NM 87106. The Altair convention is reported coming along extremely well, with thousands of people planning to attend. The dates are March 27 and 28 and late registrations may be made by calling Barbara Simms at MITS, (505) 262-1951.

Polymorphic Systems has also outgrown its old facility and has moved to 737 So Kellogg, Goleta CA 93017 (mailing address: PO Box 2207, Goleta CA 93018). Richard Peterson reports that their video board is doing well and they have just come out with a small two board, Altair compatible microprocessor (see ad page 73).

The Micro-Altair, shown here, uses an 8080 chip and Altair-compatible backplane. The standard configuration includes a CPU board with 512 bytes RAM, operating system ROM and space for more ROM, video interface board, backplane with power supply and the cabinetry shown. Price of the system is $582 with a special introductory offer of $475 through the Ides of March (March 15 1976).

8080 SOFTWARE BOARD

###ALTAIR 8800 AND IMSAI PLUG-IN COMPATIBLE. ###GIVES YOU EDITOR, ASSEMBLER, AND MONITOR ROUTINES IN A MEMORY THAT DOESN'T FORGET -- THANKS TO OUR PRE-PROGRAMMED EROMS. ###ALLOWS FOR CHANGE: BECAUSE THIS IS EROM RATHER THAN MASK, CHANGES CAN BE MADE AT WILL IN THE SOFTWARE...THE BOARD DOESN'T BECOME OBSOLETE AS YOUR SYSTEM GOES THROUGH CHANGES. ###THE COST IS $159.95 FOR THE BOARD, PARTS, SOCKETS FOR ALL ICS, REGULATORS, SOURCE LISTING, INSTRUCTIONS, ETC. ###ALSO: SOURCE LISTING ALONE AVAILABLE FOR $2.95.

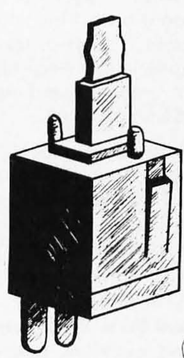
GODBOUT
BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

YOU MAY PLACE MASTERCHARGE® OR BANKAMERICARD® ORDERS BY CALLING, 24 HOURS A DAY,
(415) 357-7007

TERMS: ADD 50¢ TO ORDERS UNDER $10. ADD SHIPPING WHERE INDICATED; OTHERWISE, ITEMS SHIPPED POSTPAID. CAL RES ADD TAX. NO COD---IT'S TOO MUCH PAPERWORK!

CHERRY Switches and keytops





Cherry Stock Number M61-100

THE LIST PRICE OF THESE QUALITY DATA ENTRY SWITCHES IS $1.25 EACH---BUT OUR PRICE: 5/$1.00 (WITHOUT KEYTOPS). IF YOU'D LIKE SWITCHES WITH KEYTOPS, THE PRICE IS 5/$1.25; BUT WE ONLY HAVE A LIMITED NUMBER OF KEYTOPS, AND CANNOT GIVE CHOICE OF LEGEND. HOWEVER, THIS SWITCH IS COMPATIBLE WITH MANY COMMERCIALLY AVAILABLE KEYTOPS. HERE'S YOUR CHANCE TO STOCK UP ON SOME SMOOTH AND RELIABLE SPST PUSHBUTTON SWITCHES AT A SUPER SPECIAL PRICE.

PUSHBUTTON SWITCHES 4 FOR $1.00 SPST MOMENTARY



CAVE grafix

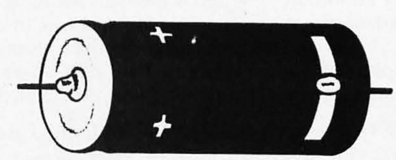
39µf tantalums for bypassing



4 for $1.00

10000 uF at 10V! $1.25

THE PERFECT TONIC FOR POWER SUPPLIES WITH A FARAD DEFICIENCY. AXIAL LEAD PACKAGE



NAKED RAM 4K x 8 $79.95

THIS GENERAL PURPOSE BOARD IS DESIGNED SPECIFICALLY FOR JOLT SYSTEMS, BUT IS EQUALLY APPLICABLE TO OTHER BI-DIRECTIONAL BUSS SYSTEMS. IF YOU DON'T NEED THE ON BOARD REGULATION OR BUFFERS OF OUR "BIGGER BROTHER" 4K x 8 BOARD, THEN THIS IS THE WAY TO IMPLEMENT COST-EFFECTIVE MEMORY. LOW POWER --- UNDER 750 MA. SAME SIZE AS JOLT RAM CARD; WITH INSTRUCTIONS.

91L02 Low Power NMOS static 1K RAMS

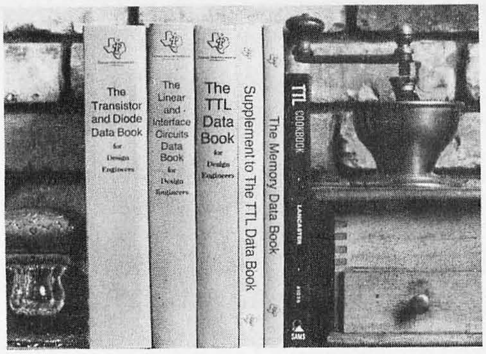
WHILE THEY LAST! SPECIAL! $2.35

micro-computer power supply $44.95

5 VOLTS---5 AMPS
+12 VOLTS---½ AMP
-12 VOLTS---½ AMP
--AND ONE ADJUSTABLE NEGATIVE BIAS SUPPLY

Ideal for systems built around 8008, 8080, PACE, 6800, 6502 and other CPUs---with enough left over to power up to 16K bytes of RAM. Same size as JOLT supply. Just about goof proof, too; includes crowbar overvoltage protection on +5V & FOLDBACK current limiting --- regulation on the 5 volt supply is guaranteed better than 1% with line and load variations. A TOP QUALITY COMPUTER SUPPLY...INCLUDES CHASSIS AND ALL HARDWARE, but less line cord. For 117 VAC, 60 Hz.

PLEASE ADD SHIPPING FOR 8 LBS.



You'll Want to Nybble at these Byte Books

Where does the editor of a computer magazine turn to when he must verify some author's hardware design? Information on a 75450 interface gate, or a 74147 priority encoder circuit does not spring forth by magic. Checking the information supplied by authors is part of BYTE's quality control program.

When you build a project, you need this same sort of information. All you find in the advertisements for parts are mysterious numbers identifying the little beasties ... hardly the sort of information which can be used to design a custom logic circuit. You can find out about many of the numbers by using the information found in these books. No laboratory bench is complete without an accompanying library shelf filled with references – and this set of Texas Instruments engineering manuals plus Don Lancaster's *TTL Cookbook* will provide an excellent starting point or addition to your personal library.

● **The TTL Cookbook** by Don Lancaster, published by Howard W. Sams, Indianapolis, Indiana. Start your quest for data here with Don's tutorial explanations of what makes a TTL logic design tick 335 pages, $8.95 postpaid.

● **The TTL Data Book for Design Engineers,** by Texas Instruments Incorporated. How does an engineer find out about the TTL circuits? He reads the manufacturer's literature. This 640 page beauty covers the detailed specs of most of the 7400 series TTL logic devices. No experimenter working with TTL has a complete library without **The TTL Data Book for Design Engineers.** Order yours today, only $3.95 postpaid.

● **The Supplement to The TTL Data Book for Design Engineers,** by Texas Instruments Incorporated. What happens when you can't find a 7400 series device listed in **The Data Book for Design Engineers?** Before you start screaming and tearing your hair out in frustration, turn to the Supplement. The Supplement has 400 pages of additional information including a comprehensive index to both TTL Data Book volumes. To be complete (and keep your hair in place and vocal cords intact) you'd best order the supplement at $1.95 to accompany the main volume.

● **The Linear and Interface Circuits Data Book for Design Engineers,** by Texas Instruments Incorporated. When you run across one of those weird numbers like 75365 the immediate frustration problem occurs again. What kind of gate could that be? We won't tell in this ad, but you can find out by reading the specifications in **The Linear and Interface Circuits Data Book for Design Engineers.** You can interface your brain to the 72xxx (linear) and 75xxx (interface) series of functions by ordering your copy of this 688 page manual at only $3.95 postpaid.

● **The Semiconductor Memory Data Book for Design Engineers,** by Texas Instruments, Incorporated. Don't forget the importance of memories to your systems. Refer to this 272 page manual to find out about the T.I. versions of many of the popular random access memories and read only memories. Order your personal copy today, only $2.95 postpaid.

● **The Transistor and Diode Data Book for Design Engineers,** by Texas Instruments, Incorporated. You'd expect a big fat data book and a wide line of diodes and transistors from a company which has been around from the start of semiconductors. Well, its available in the form of this 1248 page manual from T.I. which describes the characteristics of over 800 types of transistors and over 500 types of silicon diodes. This book covers the T.I. line of low power semiconductors (1 Watt or less). You won't find every type of transistor or diode in existence here, but you'll find most of the numbers used in switching and amplifying circuits. Order your copy today, only $4.95 postpaid.

● **The Power Semiconductor Handbook for Design Engineers** by Texas Instruments, Incorporated. To complement the low power transistor handbook, T.I. supplies this 800 page tome on high power transistors and related switching devices. Here is where you find data on the brute force monsters which are used to control many Watts electronically. Fill out your library with this book, available for only $3.95 postpaid.

● **Understanding Solid State Electronics** by Texas Instruments, Incorporated. This is an excellent tutorial introduction to the subject of transistor and diode circuitry. The book was created for the reader who wants or needs to understand electronics, but can't devote years to the study. This 242 page softbound book is a must addition to the beginner's library at only $2.95.

● **The Optoelectronics Data Book for Design Engineers** by Texas Instruments, Incorporated. This 366 page book is a compendium of information on T.I. phototransistors, LEDs and related devices. Order yours at $2.95 postpaid.

Buyers of these books should be cautioned: heavy reading will be required. These books are so filled with information that they weigh in at a total of about 190 ounces (5387 grams). On the basis of sheer mass, these books have got to be the bargain of the century. Make sure that you use a structurally sound book shelf and above all avoid dropping one of these books on your foot. But the mass of these books doesn't affect the bargain: we pay postage on all orders shipped to addresses in the USA and Canada, so the prices you see are the prices you pay. (That's only $.005 per gram on the average.)

____ TTL Cookbook @ $8.95
____ TTL Data Book @ $3.95
____ Supplement to TTL Data Book @ $1.95
____ Linear and Interface Circuits @ $3.95
____ Semiconductor Memory Data @ $2.95
____ Transistor and Diode Data Book @ $4.95
____ Understanding Solid State Electronics @ $2.95
____ Optoelectronics Data Book @ $2.95
____ Power Semiconductor Handbook @ $3.95

Send to: Name ________________

Address ________________

City ________ State ________ Zip ________

☐ Check enclosed

☐ Bill MC #________ Exp. Date ________

☐ Bill BA #________ Exp. Date ________

Signature ________________

BYTE PETERBOROUGH, NH 03458

Feel free to photocopy this or any other page if you wish to keep your BYTE intact.

ATTENTION All Ye Alice Freaks and Other Lovers of Logical Systems. Here is documented evidence that Lewis Carroll would have read BYTE had he lived in 1976:

JABBERWOCKY.

'Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe;
All mimsy were the borogoves,
  And the mome raths outgrabe.

"Beware the Jabberwock, my son!
  The jaws that BYTE, the claws that catch!
Beware the Jubjub bird, and shun
  The frumious Bandersnatch!"

He took his vorpal sword in hand:
  Long time the manxome foe he sought–
So rested he by the Tumtum tree,
  And stood awhile in thought.

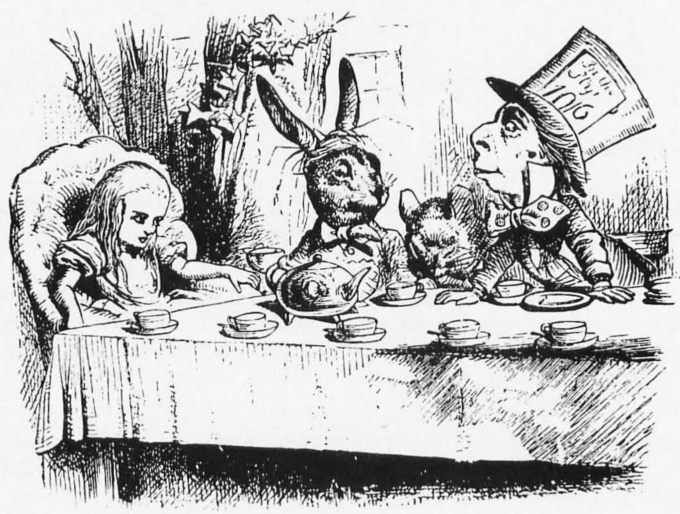
And as in uffish thought he stood,
  The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
  And burbled as it came!

One, two! One, two! And through and through
  The vorpal blade went snicker-snack!
He left it dead, and with its head
  He went galumphing back.

"And hast thou slain the Jabberwock?
  Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
  He chortled in his joy.

'Twas brillig, and the slithy toves
  Did gyre and gimble in the wabe;
All mimsy were the borogoves,
  And the mome raths outgrabe.

Don't miss out on all the fun and high quality information which is found in every issue of BYTE. Subscribe today. Join the tea party and have a BYTE to eat.



If the subscription card is missing from your issue, use the coupons below.

Name ....
Address ....
City .... State.......... Zip......
☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

Name ....
Address ....
City .... State.......... Zip......
☐ BILL ME ☐ Check for $12 enclosed
☐ Bill BankAmericard or MasterCharge # ....

• BYTE • 70 Main St • Peterborough NH 03458

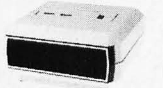
Allow six weeks for processing.

CRYSTALS

Part #	Frequency	Case Style	Price
CY1A	1.000 MHz	HC33/U	$4.95
CY2A	2.000 MHz	HC33/U	$4.95
CY3A	4.000 MHz	HC18/U	$4.95
CY7A	5.000 MHz	HC18/U	$4.95
CY12A	10.000 MHz	HC18/U	$4.95
CY19A	18.000 MHz	HC18/U	$4.95
CY22A	20.000 MHz	HC18/U	$4.95
CY30B	32.000 MHz	HC18/U	$4.95

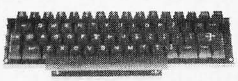
— AVAILABLE IN THESE FREQUENCIES ONLY —

CLOCK CASES



Nicely styled cases complete with red bezel for use in such applications as desk clocks, car clocks, alarm clocks, instrument cases.
DIMENSIONS: W-4'', L 4½'', H-2''.
$5.95

64 Key Keyboard $24.95



This keyboard is composed of 64 Magnetic Reed Switches, in one molded unit. It is unencoded with each SPST switch brought out to two pins.

HD0165 Keyboard Encoder ROM **$7.95**

DIGITAL WATCH READOUT

FOR USE WITH WATCHES, DVMS, COUNTERS, ETC. **$1.95**



MOS LSI DEVICES

CLOCK CHIPS

MM5309	6 Digit, BCD Outputs, Reset PIN.	$5.95
MM5311	6 Digit, BCD Outputs, 12 or 24 Hour	4.95
MM5312	4 Digit, BCD Outputs, 1 PPS Output	4.95
MM5313	6 Digit, BCD Outputs, 1PPS Output	4.95
MM5314	6 Digit, 12 or 24 Hour, 50 or 60 Hz	4.95
MM5316	4 Digit, Alarm, 1PPS Output	6.95
MM5318	Video Clock Chip, For Use With MM5841	9.95
CT7001	6 Digit, Calander, Alarm, 12 or 24 Hour	6.95

CALCULATOR CHIPS

MM5725	6 Digit, Four Function, Less Decimal	$2.95
MM5738	8 Digit, 5 Function; +, −, ×, ÷, %	2.95
MM5739	8 Digit, 4 Function, Floating Decimal	2.95
CT5001	12 digit 4 Function	3.95
CT5005	12 digit 4 Function with Memory	5.95
CT5030	12 digit 4 Function and %	7.95

MISC. MOS

MK5007	Complete 4 Digit Counter	$10.95
LD110/LD111	3½ Digital Voltmeter Chip Set	$25.00
MC1408L7	7 Bit Digital To Analog Conv.	9.95
MM5320	TV Camera Sync Generator	19.95
MM5841	Video Generator For MM5318	18.00

DL728

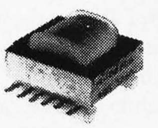
The DL728 is a dual 0.5'' common cathode red display. It is ideal for use with clock chips, as segments are already multiplexed. **$2.95**



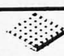
1¼''x1½'' XFMERS P.C. Mount

That were designed for clock type applications. 110 Vac primary @ 60 Hz.
Secondaries: 8-10 Vac @ 30 mA-50 mA
50 Vac @ 30 mA-50 mA
Excellent for miniature power supplies & gas discharge displays

SPECIAL $.79



1/16 VECTOR BOARD



0.1'' Hole Spacing P-Pattern

	Part No.	Length	Wideth	Price 1-19	Price 20-49
PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 02XXXP	4.50	17.00	3.69	3.32
EPOXY GLASS	64P44 062	4.50	6.50	2.07	1.86
	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

VECTOR WIRING PENCIL



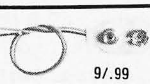
Vector Wiring Pencil P173 consists of a hand held featherweight (under one ounce) tool which is used to guide and wrap insulated wire, fed off a self-contained replaceable bobbin, onto component leads or terminals installed on pre-punched "P" Pattern Vectorbord®. Connections between the wrapped wire and component leads, pads or terminals are made by soldering. Complete with 250 FT of red wire.
Special $7.95

REPLACEMENT WIRE — BOBBINS FOR WIRING PENCIL

W36-3-A-Pkg. 3	250 ft. 36 AWG GREEN	$2.40
W36-3-B-Pkg. 3	250 ft. 36 AWG RED	$2.40
W36-3-C-Pkg. 3	250 ft. 36 AWG CLEAR	$2.40
W36-3-D-Pkg. 3	250 ft. 36 AWG BLUE	$2.40

9V BATTERY CLIP

STANDARD CLIP FOR USE WITH 9V TRANSISTOR BATTERIES WITH 4'' LEADS **9/.99**



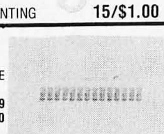
TERMINAL STRIPS

THREE TERMINAL STRIPS, WITH CENTER TERMINAL USED FOR MOUNTING **15/$1.00**



MOLEX PINS

PINS USED AS INEXPENSIVE SUBSTITUTE FOR SOCKETS
SPECIAL 100/1.49
SPECIAL 1000/12.00



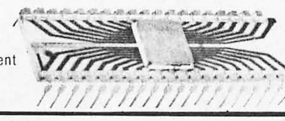
MICROPROCESSOR COMPONENTS

CENTRAL PROCESSOR UNITS

Each processor feature a 2μs instruction cycle, and are brand new from the manufacturer. The 8080 is a 2nd generation processor and the 8080A is a 2½ generation microprocessor.

8080 $24.95 — Direct Replacement For Intel C8080

8080A $39.95 — Direct Replacement For Intel C8080A



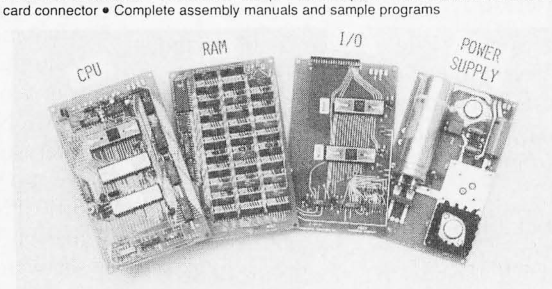
CPU'S			RAM'S			
8008	8 Bit CPU	$19.95	1101	256X1	STATIC	$2.25
8080	Super 8008	$24.95	1103	1024X1	DYNAMIC	2.95
8080A	Super 8008	$39.95	2101	256X1	STATIC	6.95
			2102	1024X1	STATIC	2.49
SR'S			2107	4096X1	DYNAMIC	19.95
2504	1024 DYNAMIC	$9.00	2111	256X4	STATIC	7.95
2518	HEX 32 BIT	7.00	7010	1024X1	MNOS	29.95
2519	HEX 40 BIT	4.00	7489	16X4	STATIC	2.49
2524	512 DYNAMIC	2.95	8101	256X4	STATIC	7.95
2525	1024 DYNAMIC	6.00	8111	256X4	STATIC	7.95
2527	DUAL 256 BIT	3.95	8599	16X4	STATIC	3.49
2529	DUAL 512 BIT	4.00	91L02	1024X1	STATIC	2.75
2532	QUAD 80 BIT	3.95	74200	256X1	STATIC	6.95
2533	1024 STATIC	7.95	93410	256X1	STATIC	1.75
3341	FIFO	6.95	5262	2048X1	DYNAMIC	2.95
74LS670	16X4 REG.	3.95				
UART'S			**PROMS**			
AY-5-1013	20K BAUD	$6.95	1702A	2048	FAMOS	15.95
ROM'S			5203	2048	FAMOS	14.95
2513	CHAR. GEN.	11.00	82S23	32X8	OPEN C.	5.00
7488	RANDOM BITS	3.50	82S123	32X8	TRISTATE	5.00

2K RAM SPECIAL

MM5262 Fully decoded 2Kx1 dynamic RAM. All inputs except clocks are TTL compatible. Provides a 635 n s minimum access time, and requires a +5, +8.5, and −15V power supply. Low power provides non volatile memory using battery back up.
$0.99 ea (0.05 cents per bit)

BUILD YOUR OWN JOLT MICROCOMPUTER IN JUST 3 HOURS OR LESS FOR $159.95.

A COMPLETE MICROCOMPUTER IN A SINGLE CPU KIT INCLUDES: • An MOS Technology MCS6502 NMOS microprocessor • 512 bytes of program RAM, and 64 bytes of interrupt vector RAM • 1K bytes of mask programmed ROM containing DEMON, a powerful debug monitor • 26 programmable I/O lines • Internal RC clock, or crystal controlled clock with user supplied crystal • Serial I/O ports for use with a teleprinter current loop drive/receiver, or an EIA standard driver/receiver • Expandable address and data buses • Hardware interrupt • Control panel interface lines available on card connector • Complete assembly manuals and sample programs



JOLT ACCESSORY KITS

JOLT RAM Card — Fully static 4,096 bytes of RAM with 1 microsecond access time and on-board decoding. **$199.95**

JOLT I/O Card (Peripheral Interface Adapter) — 2 PIA LSI chips, 32 I/O lines, four interrupt lines, on-board decoding and standard TTL drive. Fully programmable. **$95.50**

JOLT Power Supply — Operates at +5, +12 and −10 voltages. Supports JOLT CPU, 4K bytes of RAM and JOLT I/O card — or, CPU and 8 I/O cards. **$99.95**

JOLT +5V Booster Option — Fits onto JOLT Power Supply card. Supports CPU, 8K bytes RAM and 8 I/O CPU and cards. **$24.95**

JOLT Universal Card — Same size (4¼'' x 7''), same form factor as other JOLT cards. Completely blank, drilled to accept 14, 16, 24 or 40 pin sockets. **$24.95**

JOLT Accessory Bag — Contains enough hardware to connect one JOLT card to another, flat cable, connectors, card spacers, hardware, etc. **$39.95**

JOLT Resident Assembler — Fully symbolic, single pass resident assembler, all mnemonics compatible with timesharing assemblers. Delivered on four 1702A PROMs, ready for plugging into JOLT PROM card. **$149.95**

JOLT 1702A PROM Card — Sockets for 2,048 bytes of PROM memory. Place anywhere in memory with jumper selectable addresses. **$99.95**

30 AWG WIRE WRAP WIRE

AWG	COLOR	25 FT. MIN.	50 FT.	100 FT.	1000 FT.
30 AWG	WHITE	$2.10	$2.75	$3.50	$24.00
30 AWG	YELLOW	2.10	2.75	3.50	24.00
30 AWG	RED	2.10	2.75	3.50	24.00
30 AWG	GREEN	2.10	2.75	3.50	24.00
30 AWG	BLUE	2.10	2.75	3.50	24.00
30 AWG	BLACK	2.10	2.75	3.50	24.00

26 AWG RIBBON CABLE



CONDUCTORS	1-9 ft	10-24 ft	25-99 ft	100 ft
10	.39	.29	.25	.23
20	.59	.49	.45	.43
26	.69	.59	.55	.52



6' 2 CONDUCTOR POWER CORDS 125V @ 5A



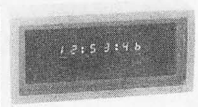
3/1$

THREE CONDUCTOR POWER SUPPLY CORDS

NUMBER	LENGTH	AWG	O.D.	RATING	COLOR	PRICE
17236	6	18(41X34)	.265	1250W 10A-125V	BLACK Special	.79 ea.
17237	6	18(41X35)	.253	1250W 10A-125V	GREY	1.25
17238	8	18(41X34)	.265	1250W 10A-125V	BLACK Special	.99 ea.
17239	8	18(41X34)	.253	1250W 10A-125V	GREY	1.30
17000	4	18(41X34)	.253	1250W 10A-125V	BLACK	.59

JE SERIES KITS

JE700 CLOCK



The JE700 is a low cost digital clock, but is a very high quality unit. The unit features a simulated walnut case with dimensions of 6'' x 2½'' x 1''. It utilizes a MAN72 high brightness readout, and the MM5314 clock chip.

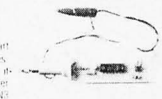
12 or 24 Hour

115 VAC—

$17.95

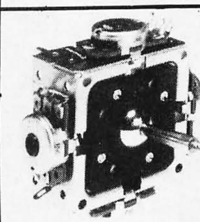
JE803 PROBE

The Logic Probe is a unit which is for the most part indespensible in trouble shooting logic families TTL, DTL, RTL, CMOS. It derives the power it needs to operate directly off of the circuit under test, drawing a scant 10 mA max. It uses a MAN3 readout to indicate any of the following states by these symbols: (HI) - 1 (LOW) - 0 (PULSE) - P. The Probe can detect high frequency pulses to 45 MHz. It can't be used at MOS levels or circuit damage will result.



$9.95 Per Kit
printed circuit board

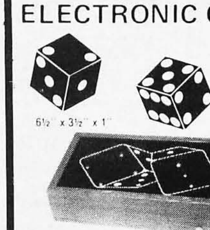
JOYSTICK



These joysticks feature four 100K potentiometers, that vary resistance proportional to the angle of the stick. Sturdy metal construction with plastics components only at the movable joint. Perfect for electronic games and instrumentation.

$9.95 ea

ELECTRONIC CRAPS

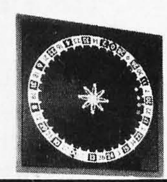


6½'' x 3½'' x 1''

Electronic craps is an entirely electronic game featuring a dice "roll down". 14 LED's form two dice that roll when activated by a push botton. Dimensions are 6½'' x 3½'' x 1½''.

$19.95

ELECTRONIC ROULETTE



This kit comes complete with all components, including the case and the cord. Electronic Roulette, is entirely electronic, and features 32 LED's that form a wheel, that is activated, by a push botton switch. Dimensions are 6½'' x 6½'' x 1½''.

$29.95

Semiconductor Specials

2N2222 - NPN Si, High Frequency switching transistor. (typically 250 MHz), 0.5W, HFe = 100, Vceo 30V. **7/$1.00**

MJ3055 - Same as TO3 CAN 2N3055 except that base and emmitter leads have solder lugs. **$.59**

1N4308 - Excellent signal diode for uses in digital switching circuits 80 PRV @ 200 mA. **20/$1.00**

MCT2E - NPN Opto Isolator which provides 2500 VDC isolation. **$2.95**

2N4123 - NPN Si, high frequency switching transistor. With an HFE = 120, it is ideally suited for most digital applications. **8/$1.00**

RL4403 - Red .2'' LED with 2 piece plastic mounting **3/$1.00**

74S287 - Outstanding 256 x 4 TTL PROM. Tristate outputs provide easy bussing. Access time of 40 ns **$7.95**

CA3081 - Common Emitter H.V. 7 transistors in a 16 PIN package. Ideal for segment drivers. **$1.75**

CA3082 - Common Collector H.V. NPN transistor array, packaged 7 transistors to each D.I.P. Excellent in uses with incandescent displays. **$2.00.**

Continental Specialties



PROTO BOARD 6

The PB-6 lets the user test and build circuits without soldering or patch cords: all interconnections between components are made with common #22 AWG hook-up wire. This quality breadboarding kit includes 630 component tie points at less than 2.5c each. It measures 6'' long by 4'' wide. Designed specially to Breadboard Microprocessor Circuits.

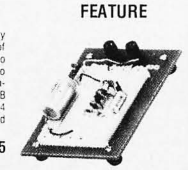
$15.95

FEATURE

PROTO BOARD 100

A low cost, big 10 IC capacity breadboard kit with all the quality of QT sockets and the best of the Proto Board series . . . complete down to the last nut, bolt and screw. Includes 2 QT-35S Sockets; 1 QT-35B Bus Strip; 2 5-way binding posts; 4 rubber feet, screws, nuts, bolts; and easy assembly insturctions.

SPECIAL $17.95



Satisfaction Guaranteed. $5.00 Min. Order. U.S. Funds.
California Residents — Add 6% Sales Tax — Data Sheets 25c each
Send a 13c Stamp (postage) for a FREE 1976 Catalog

JAMES

P.O. BOX 822, BELMONT, CA. 94002
PHONE ORDERS — (415) 592-8097

Clubs and Newsletters



Mike and Key Amateur Radio Club

The Mike and Key Radio Club, Seattle WA, has had information and activities related to microprocessors. The March 6 meeting included demonstrations of systems by four members using various Teletype models and CRT terminals. For more information, contact Bill Balzarini, K7MWC, at RO-2-7738.

Chess Interest?

J P Summerville, 2822 S Moreland, Apt 3, Cleveland OH 44120 (216) 921-1103, is interested in correspondence with other individuals engaged in activities related to programming of Chess automatons.

Little Rock Club Activities?

David W Davis, Rt 7 Box 5068-10, Benton AR, is interested in starting a club in the Little Rock area. David wants to meet other individuals with similar interests (microcomputers and amateur radio) and would like to share equipment and experiences.

St Petersburg FL Club Activity?

Allen Swann, 2510 Oak Trail S, Clearwater FL 33516, wants to help get a computer club started in the St Petersburg area. He can be reached by phone at (813)535-4194.

The Micro-8 Computer User Group Newsletter

Hal Singer and John Craig continue to put out an excellent compendium of reader supplied information in the Micro-8 newsletter. As April BYTE goes to press in late January, we received the Volume 2 Number 1 issue. Hal and John have one of the oldest hobbyist newsletters with a true microcomputer orientation, started soon after the landmark 1974 article in *Radio Electronics* which described an 8008 processor available in an extremely economical format, the Mark 8.

The format of the Micro-8 is largely oriented to making available copies of camera ready materials sent in by readers and friends. The current issue (dated January 11), for instance, includes some notes by John Craig, letters from users of various systems, names and addresses of individuals interested in contacting others, frank comments about gripes and grievances, technical comments, diagrams and programs for various systems, etc. You'll find out best about what's in it by subscribing to this pioneer newsletter, at $6 for 6 issues. Back issue sets are also available: Volume 1, 1 through 4, is $3.50; Volume 1, 5 through 12, is $6. Send to:

> *Micro-8 Computer User Group Newsletter*
> John Craig and Hal Singer, Editors
> Cabrillo Computer Center
> 4350 Constellation Rd
> Lompoc CA 93436

Tallahassee Amateur Computer Society

BYTE received a copy of the Tallahassee Amateur Computer Society Newsletter, Volume 1 Number 0 (December 1975). This was a one sheet preliminary edition of the TACS Newsletter. TACS is a group of people interested in computers as a hobby. Meetings are biweekly on Saturdays from 2:00 to 5:00 PM. Anyone with an interest or information to contribute in areas of hardware, software and applications of these wonderful machines is urged to attend.

There are varying levels of expertise among members. Several members are professional programmers or electronics engineers. Some members are new to both areas. A few members are constructing their own microcomputer systems at home or at work. Use of test equipment and consulting is available on a limited basis.

One function of the group is a stockpiling of pertinent information. A small reference library has been started, and items are available to all members for short term loan or copying. At present this includes several hobbyist magazines; detailed information on Intel 8080 systems, Motorola 6800 systems, and various interface and accessory units; a book of techniques for programming 8 bit micro systems; a programmed course in microcomputer architecture; and various works on writing editors, assemblers and compilers. Anyone with materials to contribute to this collection is encouraged to contact the group librarian, Larry Hughes.

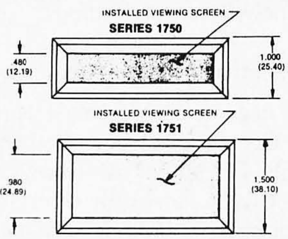
Another function of the group is to share information and techniques via seminars or tutorials. So far, topics have been programming an 8080 based system, reading a



NEW

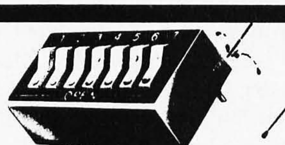
IEE-ATLAS™

Series 1750, 1751 Display Mounting Hardware



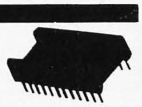
Molded socket block accepts standard 7 segment LED readouts with .3" row spacing. Pins are .65" long wire wrap type. Bezel and socket block are black molded plastic with viewing screen available in red, amber, or smoky neutral, circularly polarized for glare reduction. Unique mounting system is self fastening to panel cutout. Two sizes available. 1750 series for use with up to .4" high readouts. 1751 series for use with up to 1" high readouts.
IMPORTANT! Last 2 digits of part numbers shown below denotes number of readout positions. (specify screen color)

1750-04...$7.00	1750-06...$9.00	1750-08..$11.00
1751-04...$7.27	1751-06...$9.31	1751-08..$11.34



MINIATURE ROCKER DIP SWITCHES
Dual in-line SPST switch arrays for P.C. mounting. Spring loaded sliding ball contact system for positive, tease proof contact. Comes in contact arrangements from 4 to 10 per pack. Fits standard DIP sockets. Last two digits of stock number indicate number of switches.

DIS-76-B04....................$3.10
DIS-76-B06....................$3.50
DIS-76-B07....................$3.75
DIS-76-B08....................$3.95
DIS-76-B10....................$4.35



IC SOCKETS - HIGHEST QUALITY!!
PC Mount....Solder Tail

Skt-0802..8 pin..10/$2.25	Skt-1402..14 pin..10/$2.40
Skt-1602..16 pin.10/$2.70	Skt-1802..18 pin..10/$4.25
Skt-2202..22 pin.10/$5.50	Skt-2402..24 pin..10/$6.00

WIRE WRAP TAILS

Skt-1400..14 pin. 10/$4.50	Skt-1600..16 pin..10/$5.00

IM6100 CPU. Intersils' 12 Bit CMOS CPU chip is the microprocessor which recognizes the famous PDP8/E instruction set. Single power supply, 4-7V @ 400 microamps.
Now, a new low price.............. $52.50
Full data packet...............$4



LM317 Voltage Regulator. 1.5A, 3 terminal adjustable regulator in TO-3 case. Adjusts from +1.2V to +37V. Complete overload protection. .1% load regulation, .01% /V line regulation. No need to stock assorted regulators- just stock resistors.......................$4.99



MC 14412 UNIVERSAL MODEM CHIP
MC14412 contains a complete FSK modulator and de-modulator compatible with foreign and USA communications. (0-600 BPS)
FEATURES:
- .On chip crystal oscillator
- .Echo suppressor disable tone generator
- .Originate and answer modes
- .Simplex, half-duplex, and full duplex operation
- .On chip sine wave
- .Modem self test mode
- .Selectable data rates: 0-200, 0-300, 0-600
- .Single supply
  VDD=4.75 to 15VDC - FL suffix
  VDD=4.75 to 6 VDC -VL suffix

TYPICAL APPLICATIONS:
- .Stand alone - low speed modems
- .Built - in low speed modems
- .Remote terminals, accoustic couplers

MC14412FL $28.99
MC14412VL.................................. $21.74
6 pages of data........................ .60

MC14411 Bit Rate Generator. Single chip for generating selectable frequencies for equipment in data communications such as TTY, printers, CRTs or microprocessors. Generates 14 different standard bit rates which are multiplied under external control to 1X, 8X, 16X or 64X initial value. Operates from single +5V supply. 11.98
4 pages of data...................................40¢
Crystal for the above...........................$4.95

AMP L' ANNY Says

WHY EXPOSE YOUR BODY TO THE ELEMENTS? AVOID GETTING SOAKED BY HIGH PRICES AND MAILING FEES. SAVE TIME, MONEY, GAS AND COLD REMEDIES. BRIGHTEN UP YOUR DAY, SHOP THE TRI-TEK WAY!!!

Exclusive!

RCA GOLD CHIP

Linear Integrated Circuits

Brand new process by RCA in which the aluminum metalization has been replaced by gold. The chip is then hermetically sealed. What this means to you is unprecedented reliability and uniformity. Plastic parts that meet mil specs!!
Tri-Tek is proud to be the first to bring this new level of performance to you at SURPLUS PRICES. Why buy regrades???

CA301A..Improved, general purpose op-amp, 8 pin dip .. 59¢
CA307...Super 741 op-amp. 8 pin dip .... 52¢
CA324...Compensated quad op-amp, 14 pin dip $1.80
CA339A..Low offset quad comparator. 14 pin dip $1.59
CA741C..Famous general purpose op-amp, 8 pin dip.. . 45¢
CA747C..General purpose dual op-amp, 14 pin dip .... 82¢
CA748C..Externally compensated 741, 8 pin dip........49¢
CA1458..General purpose dual op-amp. 8 pin dip ..... 69¢
CA3401..Quad,single supply (5-18V) op amp. 14 pin... 89¢

Another super buy from RCA. CA555 timer. 8 pin dip ... 59¢

8080A

Improved performance version of the popular 8-bit microprocessor by INTEL............................$39.95



NSL4944 LED. Current regulated, universal diffused-lens red LED lamp. A GaAsP solid-state high intensity LED encapsulated in a plastic package containing a current regulating IC that provides constant intensity over a wide voltage range. 2 to 18V, AC or DC. Use for indicator lamps, optical coupling, battery charging circuits, logic probes, almost any place you need a lamp. Long life, wide angle. No series resistor needed. Typical 13mA forward current.
NSL4944....with panel mounting clip............ 89¢



Miniature PC mount rotary switches. Made by Spectrol. Only 1/2" dia, 3/16" pins for PC mounting. Top screwdriver adjust.
1 pole, six position, 69¢, 1 pole, 10 position 98¢

APRIL SPECIALS!!!
MC7805, 5V, 1A regulator, house numbered....... 99¢
MK-20 TO-3 mounting kit...................10/$2.50
Electrolytic Cap. 500ufd, 50VDC............ 10/$2.17
Miniature CTS PC pots, 1K.................. 12/ 97¢
Ceramic Base Pot arrays, assorted values....... 15/ 99¢
24 Pin DIP socket, gold, solder tail leads...... 10/$3.88
RTL Flip-Flop TO-5 metal can............... 10/..99¢
Transmitting Mica Capacitor .01ufd, 2.5KV... 5/..99¢
Koil Kord. 4 color coded conductors. 4 feet.. 2/..99¢
Midget Collapsable Antenna. 7" extended.... 3/..99¢
Heat Shrink Tubing 3/8"dia X 2' long, black 4/..99¢
Ferrite Beads. 1/8"X1/8" dia................20/..99¢
Computer Grade Cap..4000ufd/50V........... 2/$1.99
100V, 3A epoxy diodes, full leads........... 25/$1.98
1N4148 High speed signal diode (like 1N914)..20/ $1.00

TRI-TEK, INC.
6522 NORTH 43RD AVENUE,
GLENDALE, ARIZONA 85301
PHONE 602 - 931-6949

Minimum order $10 US/$15 Foreign in US Funds. All orders postage paid. Please add insurance. Master Charge and Bank America cards welcome, ($20 minimum.) Telephone orders may be placed 11AM to 5PM daily, Mon thru Fri. Call 602-931-6949. Check reader service card or send stamp for our latest flyers packed with new and surplus electronic components.

Is your organization listed here? Do you want to start a local club? BYTE wants to encourage the transfer of information to and among the practitioners of the personal information systems art. If your club or organization is not mentioned here, be sure to put a member in charge of sending us the information on your activities; if you're interested in starting a club, tell us and we'll help out by printing your name and address.

parallel ASCII keyboard, and using a cassette tape recorder for mass storage (data and/or programs). Anyone willing to conduct a session on any topic of interest to the group is encouraged to contact the group program director (temporarily Larry Hughes). Someone is needed to take on the role of program director.

For further information on TACS, contact Larry Hughes, home phone 575-4471, work phone 644-2019, or write him at Rt 14, Box 351-116, Tallahassee FL 32304.

Meetings are held in Room 101, Love Building, Florida State University campus. Dates for future meetings: March 13, 27, April 10, 24, May 8, 22, June 5, 19, July 10.

Oklahoma City Hobbyist Group

Mike O'Dell, PO Box 2891, Norman OK 73069, writes that he is trying to contact others in the Oklahoma City metropolitan area. He is with the Computer Science Department at the University of Oklahoma, and there is a nucleus of several individuals already involved in club activities. Anyone interested in this hobbyist group should contact Mike by mail or at (405)325-3866 (ICS Department) or at home (seldom) (405)364-0615.

Long Island Computer Club

The Long Island club's first meeting was held in January, and the second meeting February 20. The algorithm for meeting days is tentatively the third Friday of every month; arrangements are being made for a regular meeting place. For details, write Jerry Harrison at 36 Irene Ln, East Plainview NY 11803; or phone (516)938-6769.

New Orleans Club

If you live in the metropolitan New Orleans area and are interested in computers, you are invited to join our group. Whether your interest is hardware, software, applications, or just general interest, we welcome your input. For further details, please write or call:

> Emile Alline
> 1119 Pennsylvania Av
> Slidell LA 70458
> (504)641-2360

News From North Texas

Bill Fuller forwarded BYTE the latest issue of the *Computer Hobbyist Group of North Texas Newsletter*, dated January 1976. For individuals in the Dallas and Fort Worth areas, this is the place where your gregarious qualities can be exercised. Contact Lannie Walker, president, at (818)244-1013; Neil Ferguson at 461-2867, Ric Martin at (214)387-1945, or Bill Fuller at 641-2909. The January issue included Harold Mauch's comments on interfacing older Teletypes to computers (eg: Model 14, 15 and 19 devices with Baudot coding), comments on the 8080 versus 6800, short product reviews, short publications reviews, Mauch's notes on the provisional audio cassette interface standard (see his article in March BYTE), and the usual business stuff for which newsletters are famous.

Cache Newsletter (Chicago Area)

Cache's newsletter, Volume 1 Number 1, edited by Geoff Lowe, described the newsletter's goals. For individuals living in or near the Windy City, Cache is the central meeting place for hobbyists, holding sessions monthly northwest of the city. The Cache mailing address is PO Box 36, Vernon Hills IL 60061.

One of the most exciting activities reported in the January newsletter was Arthur Kingsworth and Ken Short's demonstration of the new E & L Instruments MD-1 microdesign system. Mr Short is with SUNY-Stony Brook, and was scheduled to discuss advanced digital design and 8080 related topics using the system; Mr Kingsworth talked about simple digital design. In phone conversation with William Precht, a member of Cache, the presentation sounded extremely valuable to small systems users. Such detailed nuts and bolts discussions by manufacturers' representatives are a big plus to any club meeting.

Beta Iota Tau

Larry Passo, secretary-treasurer of the new college fraternity Beta Iota Tau sent BYTE a copy of the preliminary bylaws of the fraternity. For the record, here is the statement of purposes and goals taken from the preliminary version:

> "The purpose of Beta Iota Tau is to provide an organization for college students interested in computer sciences. Beta Iota Tau has as its goals the fostering of brotherhood among such students and the encouragement and support of computer related activities and studies."

The organization is in the process of preparing the final bylaws. They're of course helped out by keeping the entire text of the bylaws on a computer filing system so that changes can be made using the text editor program.

Persons interested in the concept of a

college fraternity for computer science should contact Beta Iota Tau through:

Richard A Petke, chairman
Lawrence H Passo, secretary-treasurer
RHIT Box 520
Terre Haute IN 47803

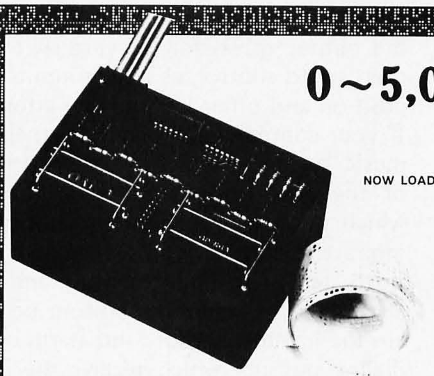
Eastern Ontario Club?

Victor Vees, 342-Palace Rd, Kingston, Ontario, K7L 4T3, is interested in forming a computer club for the Ontario-Quebec area of Canada. He can be reached by phone at (613) 546-2560.

And, From Ottawa

Larry Kayser, VE3QB/WA3ZIA, phoned BYTE recently, and described the trials and tribulations of computer hobbyists in Canada. He's interested in forming a club for the Ottawa region. Interested parties can contact him at (613)741-1640 or by mail at 24 Arundel, Ottawa, Ontario, Canada.

Right now, it looks as if the locally produced 8008 oriented products are still the most cost effective machines for Canadians. The reason? A total tariff approaching 33%. (We don't know whether it's Washington or Ottawa that causes the tariff, but it's a waste of global resources either way.)



continued from page 4

the output drives for the yard switches and train speed control, as well as inputs of train position and other layout state information. If your computer is to control an electronic music system, then the custom peripherals of the application are the output devices which are used to generate the music and the interactive control inputs. If your computer is to act as a home monitor and burglar alarm system, then the custom peripherals are the intruder sensors and alarm or phone dialing outputs which receive the information about the security status of your property.

Even if your system involves no custom hardware modules, there is always the customization of software, given a general purpose system. As soon as you have the system up and running, you'll begin to develop a personal library of programs and techniques of programming which will make your particular system of hardware plus mass storage files completely different from every other system in existence. By making a program to accomplish your specific purposes, you have made the system into a new entity which is different from another physically similar system owned by your technological neighbor. The personal library of your programming will build up on mass storage over time, further enhancing the uniqueness of your own efforts in personal computing. While you may trade programs with other individuals having similar systems, the set of programs in your own library is likely to be completely unique with respect to any other person's library. This software uniqueness, which transcends the potential sameness of hardware is one final guarantee that every system ultimately becomes a personalized system if it is used at all.

Enhancing The Options

As described above, a major component of the psychological rewards of home brew computing is the personal uniqueness of the systems which are created by BYTE readers. This form of computing is a means of creating a personally unique hardware, software and skills combination according to your own tastes and desires. This is one of the more important personal rewards to be obtained, and is an obvious motivating factor for the people who get involved in the field.

Thus one of the major issues of concern to manufacturers and users should be ways to enhance the number of options available

BYTE reader service

To get further information on the products advertised in BYTE, fill out the reader service card with your name and address. Then circle the appropriate numbers for the advertisers you select from this list. Add a 9 cent stamp to the card, then drop it in the mail. Not only do you gain information, but our advertisers are encouraged to use the marketplace provided by BYTE. This helps us bring you a bigger BYTE.

BOMB: BYTE's Ongoing Monitor Box

BYTE would like to know how readers evaluate the efforts of the authors whose blood, sweat, twisted typewriter keys, smoking ICs and esoteric software abstractions are reflected in these pages. BYTE will pay a $50 bonus to the author who receives the most points in this survey each month.

- *Articles you like most get 10 points, articles you like least get 0 (or negative) points – with intermediate values according to your personal scale of preferences, integers only.*
- *Only one entry per reader.*

Page No.	Article	LIKED LEAST ... BEST
16	Crayne: Programming the Implementation	0 1 2 3 4 5 6 7 8 9 10
20	Fox: Biorhythm for Computers	0 1 2 3 4 5 6 7 8 9 10
24	Nelson: Magic of Computer Languages	0 1 2 3 4 5 6 7 8 9 10
28	Lancaster: One Layer Printed Circuits	0 1 2 3 4 5 6 7 8 9 10
34	P. Helmers: Aargh!	0 1 2 3 4 5 6 7 8 9 10
36	Flippin: SR-52: Another World's Smallest	0 1 2 3 4 5 6 7 8 9 10
42	Bosen: Controlling External Devices	0 1 2 3 4 5 6 7 8 9 10
46	Cotton: Interface an ASCII Keyboard	0 1 2 3 4 5 6 7 8 9 10
50	Murray: Frankenstein Emulation	0 1 2 3 4 5 6 7 8 9 10
56	Wier-Brown: Design an On Line Debugger	0 1 2 3 4 5 6 7 8 9 10
64	Baker: Update: TI TMS9900	0 1 2 3 4 5 6 7 8 9 10
80	Thompson: Mini Wire Wrap	0 1 2 3 4 5 6 7 8 9 10

January BOMB Results

The leader in BOMB voting for the January 1976 BYTE was James Luscher, who receives the $50 bonus check for his article "Taking Advantage of Memory Address Space." Runners up were Sumner Loomis' "Let There Be Light Pens" and Jim Hogenson's "CT-1024 Kit Review."

to personal computing enthusiasts. One item which might be considered is the standardization of certain hardware interconnection schemes at the plug level. What I have in mind is the creation of (for example) an 8 bit bidirectional IO bus plug with control signals and timing strobes. With such a plug definition, a number of modular additions could be made independent of backplane physical and electrical considerations simply by plugging the peripheral into the standard socket. For numerous low speed peripherals, such an adapter plug for programmed IO would be ideal. I'll list a few:

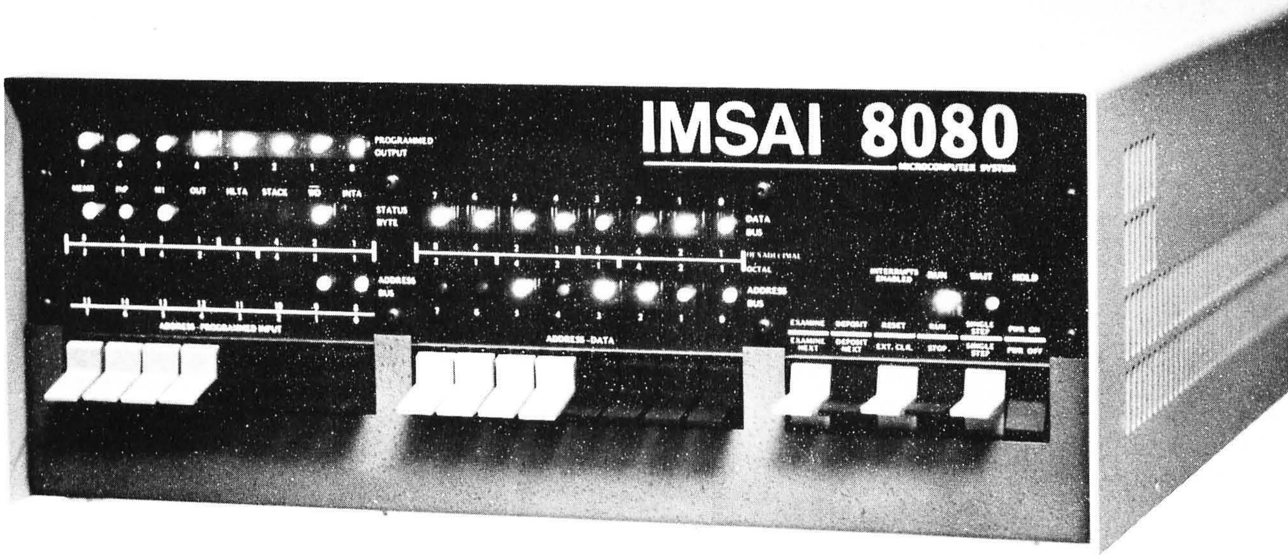
- Modular electronic musical instruments
- Low speed tape interfaces with UART serial to parallel conversion
- Digital multimeters
- Frequency counters
- Burglar and fire alarm inputs or outputs
- High fidelity system signal routing control

The idea is to achieve a standard for non-DMA programmed IO plug interfaces which could be provided as multiple plugs on an appropriate card coming with any given brand X computer. Then the manufacturers of systems for specific applications could make a "plug compatible" product for this standard which could be used with any of the processors on the market. Is there a need for such a standard? I think so, from the argument that the larger the number of options people have available, the larger will be the whole field of personal computing. Last November, BYTE organized a conference of the manufacturers and users in our infant field to discuss an audio tape recording standard. A productive result of this conference was the provisional standard described in February and March BYTEs. For the same reasons, it would be good to get the manufacturers and users together to talk about the possibility of such a programmed IO plug compatibility standard which will help expand the utility and generality of these small scale systems products.

I'd like to see what interest there is from manufacturers and users in achieving for the personal computing field the logical equivalent of high fidelity's RCA style phono plug. I think there should be enough interest to justify a working meeting next fall on the subject. The meeting will be modelled on last fall's working meeting concerning audio cassette recording standards. I invite correspondence from manufacturers and users who have something to contribute to the organization and definition of such a standard. ■

IMSAI®

COMMERCIAL GRADE COMPUTER – PRICED FOR THE INDIVIDUAL KIT BUILDER



QUALITY FEATURES INCLUDE:

A commercial grade cabinet, made of heavy gauge aluminum, custom designed for this specific product. Rack mounting is optional.

Lucite display panel with a photographically reproduced legend and LED display mask. In addition to standard lights and switches, the heavy duty front panel has an extra 8 program controlled LED's.

Rugged commercial grade paddle switches backed up by reliable debouncing circuits.

Gold plated contacts on board edge connectors to assure corrosion free contact at all locations.

A heavy duty power supply to handle lots of extra cards, the power supply delivers up to 20 amperes at 8 volts, and 3 amps each at +16 and -16 volts. A 30 ampere supply is optional.

A sturdy card cage to properly support up to 22 printed circuit boards.

Plug removable front panel board that eliminates the normal wiring harness between front panel and back plane. For special dedicated uses, the front panel board may be removed from the unit.

A full line of peripheral devices to build truly powerful and complete systems.

An exclusive shared memory facility from the company that pioneered shared memory microprocessing. Shared memory facilities allow multiple microprocessors in the same cabinet to share the same memory, yet run different programs in parallel with each other.

A full compliment of board options.

BASIC COMPUTER INCLUDES: 8080A Processor Board, Front Panel Contro‌ı ʙoard, lights and switches, power supply, Expander Board, Cabinet . . . and . . . Software (Assembler, Monitor, Text Editor, Loader and 4K BASIC).

PRICES: KIT . . . $499 ASSEMBLED . . . $931 4K RAM KIT . . . $165 ASSEMBLED . . . $299

OPTIONS: Rack mount, RAM & PROM boards, Parallel & Serial I/O Boards, an 8 level Priority Interrupt/Clock Board, Shared Memory Boards, Interface Boards.

PERIPHERALS: Floppy Disk Drives, 80 Column/300 LPM Printer, 50 Megabyte Disk Drive, 30 character per second Printer with Plotting Mode.

TERMS: Check or Money Order, Bankamericard, Master Charge, 25% deposit on COD orders. On all orders under $1,000, add 5% for handling. California residents add 6% tax.

SEND FOR FREE CATALOG OF IMSAI MICROCOMPUTER PRODUCTS – DEALER INQUIRIES INVITED

SPECIAL NOTICE TO ALTAIR 8800 OWNERS

If you would like to step-up to the superior quality of an IMSAI 8080, you will be pleased to know that your ALTAIR 8800 boards are "plug-in" usable—without modification—in the IMSAI 8080 cabinet. Furthermore, by acquiring IMSAI's unique Memory Sharing Facility, your ALTAIR MPU Board and IMSAI MPU Board can co-exist in the same cabinet, operate in parallel with each other, and share all memory in common. This is the technology that laid the foundation for IMSAI's powerful HYPERCUBE Computer and Intelligent Disk systems (recently featured in Computerworld, Datamation and Electronics magazines.)

IMS ASSOCIATES, INC., 1922 REPUBLIC AVENUE, SAN LEANDRO, CA 94577
(415) 483-2093

The MITS Altair 8800.



(It's showing up in some of the most unusual places.)